A PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SOLO PROGRAMADORES

IV, NÚMERO 46

975 Ptas.

NA OPCIÓN DE GESTIÓN POTENTE Y VELOZ

POSTGRES
LA BASE DE DATOS DE LINUX

TECNOLOGÍA 3D
Programación del acelerador gráfico i740

WINDOWS NT SERVER
Acceso remoto (RAS)

VISUAL BASIC
ActiveX unido a controles de datos

PROGRAMACIÓN MULTIMEDIA
DirectX con Delphi: DirectPlay

HARDWARE
Programación del módem (III)

PROGRAMACIÓN DE SOCKETS
Creación de un servidor Web con Delphi (II)

CONTENIDO DEL CD-ROM

- Netscape Messaging Server 3.5
- Netscape Directory Server 3.1
- Oracle Developer 2000 2.0
- ADA 95 GNAT 3.10
- ObjectADA Realtime Raven
- X-Win32

8 413042 303299

SÓLO PROGRAMADORES

Número 46
SÓLO PROGRAMADORES
es una publicación de
TOWER COMMUNICATIONS

Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es
Coordinador Técnico
Eduardo de Riquer
Redactora Jefe
Cristina Peña
crisp@towercom.es

Colaboradores
Enrique Díaz, Ernesto Schmitz, Chema Alvarez, Constantino Sánchez Ballesteros, Enrique de la Lastra, Jordi Agost, Juan Manuel Menéndez, Enrique de Alarcón

Maquetación
Susana Llano
Tratamiento de Imagen
María Arce Giménez

Publicidad
Erika de la Riva (Madrid)
Tel.: (91) 661 42 11
Pepín Gallardo (Barcelona)
Tel.: (93) 213 42 29 - 970 47 44 35

Suscripciones
Begoña Alvarez
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Laboratorio
Óscar Rodríguez (jefe)
oscarrf@towercom.es
Javier Amado
jamado@towercom.es
Seryicio Técnico
Oscar Casado
ocasado@towercom.es

Filmación
Megatipo
Impresión
Gráficas Reunidas
Distribución
SGEL
Distribución en Argentina / Chile / Colombia / México / Venezuela
Capital: Huesca y Sanabria
Interior: D.G.P.

TOWER COMMUNICATIONS
Director General
Antonio M. Ferrer Abelló
Director Financiero
Francisco García Díaz de Liaño
Director de Producción
Carlos Peropadre
Directora Comercial
Carmina Ferrer
carmina@towercom.es

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

La revista Sólo Programadores no tiene por qué estar de acuerdo con las opiniones escritas por sus colaboradores en los artículos firmados. El editor prohibe expresamente la reproducción total o parcial de los contenidos de la revista sin su autorización escrita.

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN
COPYRIGHT 31-08-98

EDITORIAL

Libres, juntos y distribuidos

Hace pocas fechas se produjo un nuevo golpe de efecto relacionado con la denominada batalla de los navegadores, protagonizada por Microsoft y Netscape. Esta última compañía anunció que la versión 5 de su navegador sería gratuita, algo lógico según las circunstancias. Pero lo que realmente nos sorprendió a todos, fue el hecho de hacer público su código fuente. Así, se creó un punto de encuentro en Internet (www.mozilla.org) para que toda la comunidad de programadores tuviese acceso a dicho código y participase de forma activa en el desarrollo final del producto.

La verdad es que existen una serie de consecuencias más o menos inmediatas derivadas de este anuncio, aunque algunas, como la relevancia de Internet entre los programadores, resulta incuestionable. Dejando a un lado las ventajas que supone la participación activa en un desarrollo de esta envergadura, el mayor beneficio que podemos extraer los programadores se basa en la posibilidad de acceder a la las tecnologías más modernas (JavaScript, CGI, HTML, etc.), implementadas por un grupo de profesionales de este sector. Esto supone un salto cualitativo en la manera de adquirir una formación completa, compartida con otros programadores y sobre todo, basada en desarrollos tangibles. Además, siempre existe la posibilidad de utilizar los nuevos conocimientos en aplicaciones propias, lo que sin duda da una idea de la trascendencia de este tipo de noticias.

Aunque esta noticia resulta de gran interés para todos nosotros, no se trata de una iniciativa pionera, puesto que Linux es el abanderado de esta nueva y emergente forma de cooperar. Lo que empezó casi como una oposición simbólica a los grandes creadores de sistemas operativos, se ha convertido en una realidad cotidiana. De hecho, las empresas comienzan a volver la cabeza hacia el proyecto, aunque eso sí, con ciertas reticencias todavía. Esta desconfianza deriva de la propia naturaleza del producto, ya que estos proyectos tienen un carácter tan dinámico, que nunca ofrecen la sensación de finalización, y por tanto de estabilidad. En gran medida, las por otra parte necesarias versiones, releases, drivers, etc., contribuyen a que los usuarios no alcancen un punto medio entre actualidad y estabilidad. Esperemos que el esfuerzo de unos no sea utilizado por otros como trampolín para obtener beneficios mediante múltiples y variadas distribuciones, o al menos, que los unos y los otros sean una misma entidad.

Tenéis a vuestra disposición la dirección de correo electrónico de la revista: solop@towercom.es, donde podéis enviar vuestras sugerencias sobre los contenidos, los temas que queráis que incluyamos y los productos que os gustaría que distribuyéramos con el CD-ROM, que todos los meses regalamos. Sois nuestros mejores colaboradores.

SOLO PROGRAMADORES

46

6 Noticias
LAS ÚLTIMAS NOVEDADES EN LA TECNOLOGÍA INFORMÁTICA

Esta sección de nuestra revista está dedicada a todas aquellas noticias de actualidad que tienen que ver con la programación y que os interesan; lo último para estar al día.

12 Visual Basic
CREACIÓN DE CONTROLES ACTIVEX (III)

Esta tercera entrega nos enseñará a crear controles ActiveX, añadiendo así a los conocimientos de las series anteriores características relacionadas con Internet. Iremos más allá del comportamiento de dichos controles para poder unirlos con controles de datos.

18 Hardware y Tecnología 3D
PROGRAMACIÓN DEL ACELERADOR GRÁFICO INTEL 740

Las aceleradoras 2D/3D están en pleno auge gra
a su extendido uso para la creación de juegos y p
gramas multimedia. Es el momento adecuado p
acercarnos al acelerador gráfico Intel740, conocer r
de cerca las características de su chip, y su capaci
de trabajo.

40 Linux
BASES DE DATOS RELACIONALES CON POSTGRES

Un nuevo artículo de portada nos abrirá toda un abanico importante de conocimientos de este sistema de bases de datos relacional, su instalación, así como su arquitectura. Aprenderemos a trabajar con bases de datos relacionales, de muy diferentes maneras. También tendremos la ocasión de estar al tanto de las versiones de PostGres más actuales.

25 Programación Multimedia
PROGRAMACION DE DIRECTX CON DELPHI (VI)

Continuando con la serie de artículos de desarrollo multimedia, tendremos la ocasión de conocer el Interfaz DirectPlay, y cómo éste es capaz de lograr la interactividad entre varios usuarios conectados a una misma red.

35 Windows NT
ACCESO REMOTO (RAS)

Cada día es más habitual que algunos trabajadores desempeñen sus cometidos desde el propio hogar ahorrando tiempo, y ganando en comodidad. Las conexiones remotas nos ofrecen toda una inmensa gama de posibilidades en este campo.

56 Hardware
PROGRAMACIÓN DEL MÓDEM (III)

Enlazando con las anteriores entregas, este tercer artículo continúa con el programa que daba comienzo en el número anterior. Es ahora el momento de tener en cuenta la posibilidad de crear una conexión entre la línea telefónica y el módem.

72 Libros
GUÍAS MUY ÚTILES PARA TODOS LOS LECTORES

Este mes dos interesantísimas guías que nos ayudaran a desenvolvernos en los nuevos y más avanzados campos de la informática. Además de dos sugerentes manuales que pueden interesar a los aficionados al tema.

62 Programación de Sockets
CREACIÓN DE UN SERVIDOR WEB CON DELPHI (II)

Una vez aprendidas las bases fundamentales para introducirnos en la programación de un servidor Web, ahora es tiempo de conocer cómo crear un servidor de este tipo mediante Delphi, y además saber cómo envía un servidor Web las páginas HTMLque solicita un navegador.

74
CORREO DEL LECTOR

En esta sección se pretende dar respuesta a todas las dudas que puedan tener nuestros lectores, pudiendo dirigirse a los propios autores o bien la dirección de correo de nuestra publicación: solop@towercom.es.

78
CONTENIDO DEL CD-ROM

En este número de la revista, incluimos un nuevo CD-ROM con dos importantes programas como son: Aonix ObjectADA Real-time Raven y Oracle Developer/2000 v2.0. Además de éstos, podreís encontrar las fuentes de algunos de los artículos que incluimos en esta edición de la revista Solo Programadores.

Noticias

Soportará Windows Foundation Classes (WFC)

NUEVA VERSIÓN DE VISUAL J++

El líder mundial Microsoft ha anunciado la disponibilidad de la beta de la próxima versión de Visual J++ 6.0, la herramienta de desarrollo más utilizada en el mundo.

Esta nueva versión ofrece a los desarrolladores que utilizan Java habitualmente un camino más rápido para construir y desplegar soluciones cliente/servidor de alto rendimiento basadas en datos para el sistema operativo Windows y para Internet/Intranet.

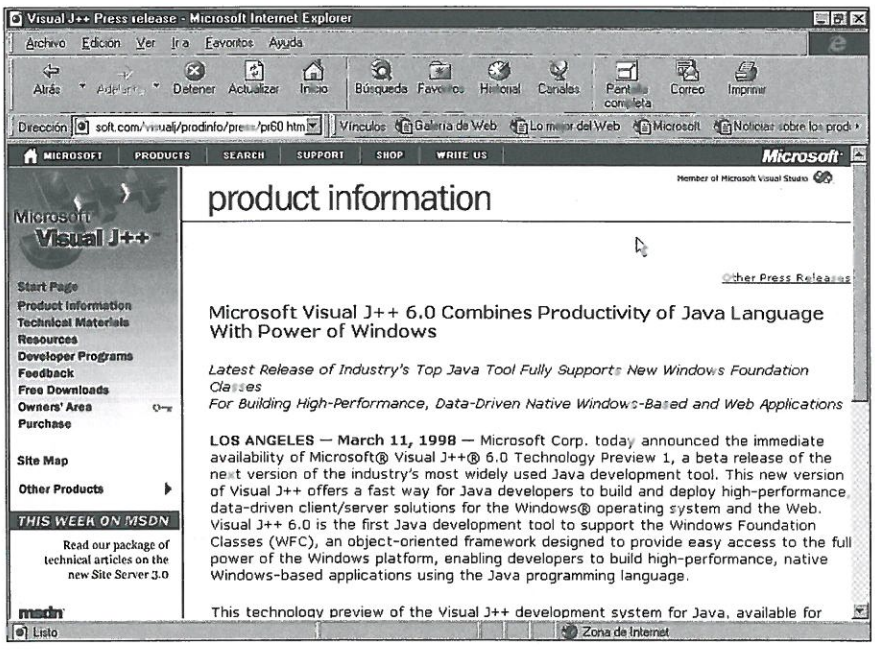
Esta versión 6.0 es la primera herramienta de desarrollo que soporta Windows Foundation Classes (WFC), un entorno orientado a objetos diseñado para obtener un rápido acceso a todas las posibilidades de plataforma Windows, permitiendo a los desarrolladores construir aplicaciones de alta calidad nativas para Windows usando el lenguaje de programación Java.

Con la intención de satisfacer la creciente demanda de desarrollo nativo en Java, Visual ++ incluye las nuevas Windows foundation Classes, que permiten a los desarrolladores crear potentes aplicaciones en Java de altas prestaciones para Windows. Las librerías de WFC unifican el desarrollo para la API Win32 y para la Web con un modelo único de programación para el cliente y el servidor.

El nuevo Visual J++ 6.0 incluye también un entorno de desarrollo integrado (IDE) actualizado y totalmente configurable, basado en el popular IDE de Visual Basic. Con él es posible crear potentes aplicaciones cliente/servidor.

Los usuarios de la anterior versión, Visual J++ 1.1, podrán disfrutar de precios especiales de actualización cuando la versión final esté disponible

En estos momentos existe una versión preliminar de Visual J++ que se puede obtener bajándola de Internet de manera gratuita en la dirección creada especialmente para ello en **www.microsoft.com/visualj/**


Visual J++ Press release - Microsoft Internet Explorer
Archivo Edición Ver Ir a Favoritos Ayuda
Atrás Detener Actualizar Inicio Búsqueda Favoritos Historial Canales Pantalla completa Correo Imprimir
Dirección soft.com/visualj/prodinfo/press/pr60.htm
Vínculos Galería de Web Lo mejor del Web Microsoft
MICROSOFT PRODUCTS SEARCH SUPPORT SHOP WRITE US
Microsoft
Member of Microsoft Visual Studio
Microsoft Visual J++
Start Page
Product Information
Technical Materials
Resources
Developer Programs
Feedback
Free Downloads
Owners' Area
Purchase
Site Map
Other Products
THIS WEEK ON MSDN
Read our package of technical articles on the new Site Server 3.0
msdn
product information
Other Press Releases
Microsoft Visual J++ 6.0 Combines Productivity of Java Language With Power of Windows
Latest Release of Industry's Top Java Tool Fully Supports New Windows Foundation Classes
For Building High-Performance, Data-Driven Native Windows-Based and Web Applications
LOS ANGELES — March 11, 1998 — Microsoft Corp. today announced the immediate availability of Microsoft® Visual J++® 6.0 Technology Preview 1, a beta release of the next version of the industry's most widely used Java development tool. This new version of Visual J++ offers a fast way for Java developers to build and deploy high-performance, data-driven client/server solutions for the Windows® operating system and the Web. Visual J++ 6.0 is the first Java development tool to support the Windows Foundation Classes (WFC), an object-oriented framework designed to provide easy access to the full power of the Windows platform, enabling developers to build high-performance, native Windows-based applications using the Java programming language.
This technology preview of the Visual J++ development system for Java, available for
Listo
Zona de Internet

Ya hay versiones beta disponibles
SE ANUNCIA EL JAVA DEVELOPMENT KIT 1.2

Sun Microsystems ha anunciado nuevos lanzamientos para el próximo Java Development Kit (JDK 1.2), que incluye mejoras en el rendimiento y una actualización completa del interfaz de programación de aplicaciones (API).

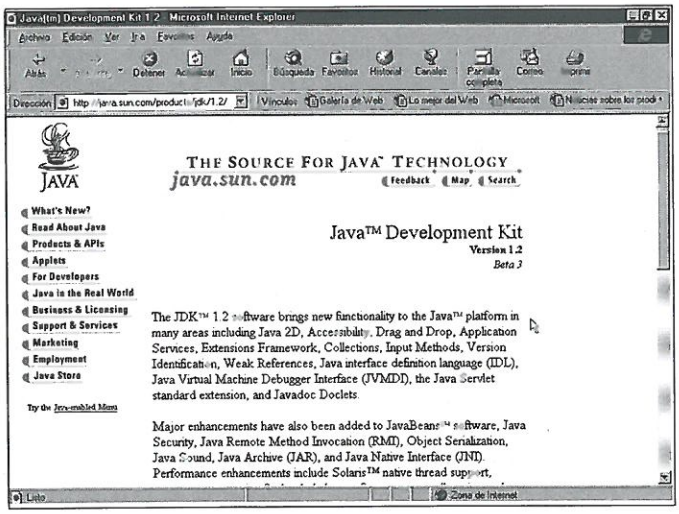
Sun ha presentado además la especificación Enterprise JavaBeans 1.0 que ha finalizado el proceso de revisión pública y ya es definitiva.

Prevista su presentación para el próximo verano, JDK 1.2 ofrece a los desarrolladores un rico conjunto de nuevas bibliotecas de clases y herramientas que facilitan más que nunca la creación de aplicaciones corporativas multiplataforma y distribuidas.

Asimismo Sun se ha comprometido a ofrecer el mayor rendimiento posible tanto a los desarrolladores como a usuarios finales de la tecnología Java.

Desde el anuncio el año pasado del JDK1.1, Sun ha mejorado de forma significativa el rendimiento de mejoras en la sincronización de los threads y de la asignación de la memoria para el entorno operativo de Sun Solaris y del ajuste de clases para todas las plataformas.

Según afirmó Jon Kannegaard, Vicepresidente de Productos de Software de la División JavaSoft de Sun "Nuestra empresa ha potenciado la plataforma Javaa en las áreas que más preocupan a los desarrolladores. Con el nuevo JDK nuestro objetivo es proporcionar el rendimiento, la fiabilidad y el conjunto de funcionalidades que los desarrolladores corparativos exigen de la plataforma estándar de Network Computing.


Java[tm] Development Kit 1.2 - Microsoft Internet Explorer
THE SOURCE FOR JAVA™ TECHNOLOGY
java.sun.com
Java™ Development Kit
Version 1.2
Beta 3
What's New?
Read About Java
Products & APIs
Applets
For Developers
Java in the Real World
Business & Licensing
Support & Services
Marketing
Employment
Java Store
The JDK™ 1.2 software brings new functionality to the Java™ platform in many areas including Java 2D, Accessibility, Drag and Drop, Application Services, Extensions Framework, Collections, Input Methods, Version Identification, Weak References, Java interface definition language (IDL), Java Virtual Machine Debugger Interface (JVMDI), the Java Servlet standard extension, and Javadoc Doclets.
Major enhancements have also been added to JavaBeans™ software, Java Security, Java Remote Method Invocation (RMI), Object Serialization, Java Sound, Java Archive (JAR), and Java Native Interface (JNI). Performance enhancements include Solaris™ native thread support,

La tercera versión beta se encuentra disponible en la URL **http://java.sun.com/products/jdk**

Los servidores incluirán aplicaciones de gestión
ALIANZA SILICON GRAPHICS-ORACLE

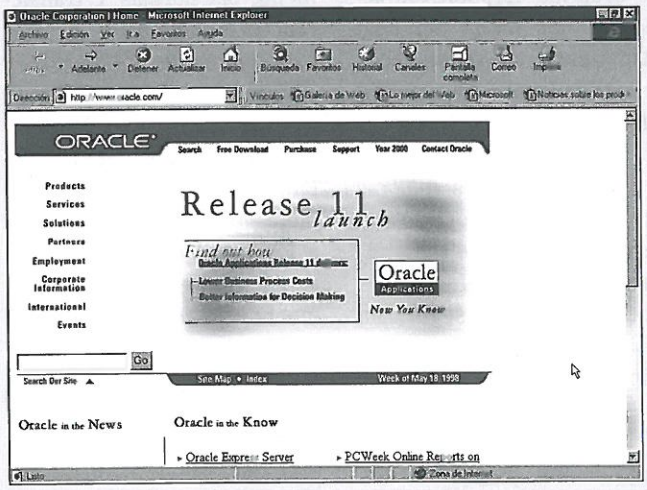
Silicon Graphics ha anunciado una alianza con Oracle, para unificar los servidores Origin200 de Silicon con las soluciones de Oracle.

Esta alianza, proporciona a las empresas funcionalidades únicas a la hora de implantar aplicaciones de gestión consideradas como críticas, y basadas en Internet, para que sean fiables, fáciles de mantener y con capacidad de crecimiento.

Silicon Graphics por primera vez, venderá y soportará las aplicaciones empresariales de Oracle para las empresas de gestión, de esta forma, SGI ofrece a los clientes la posibilidad de comprar una solución completa.

Al añadir ofertas de cliente/servidor, base de datos y software empresarial a la familia de servidores Origin200, los usuarios pueden seleccionar desde una amplia gama de opciones y obtener soporte de un sólo fabricante. Además los servidores Origin200 proporcionan la funcionalidad de alta velocidad y capacidad de crecimiento requerido por las aplicaciones de alto ancho de banda en entornos de creación Web, comercio electrónico, servidor de aplicación, y de base de datos.

Puede ampliar información en la página Web principal de Silicon Graphics. **www.sgi.com**



Herramientas abiertas y JavaServer Engine
NUEVOS PRODUCTOS PARA EL DESARROLLO DE APLICACIONES JAVA

Con la intención de permitir a los programadores profesionales y a fabricantes de software la creación e implementación de aplicaciones críticas, la compañía Sun ha anunciado un paquete de herramientas abiertas y JavaServer Engine.

Este paquete de herramientas, que se puede combinar fácilmente con los entornos integrados de desarrollo más populares, incluirá inicialmente Java Blend para automatizar la integración de aplicaciones Java con información de bases de datos relacionales; Java Modeler, que facilita el modelado y diseño de aplicaciones, y la generación y extracción de diseños de objetos de código ya existente; y JavaSafe, un sistema de control de versiones y revisiones en red para código fuente y contenidos Web.

Con estas herramientas, Sun sigue ofreciendo a los desarrolladores lo que necesitan para hacer frente a los exigentes requisitos de proyectos de los entornos corporativos actuales.

Además de este paquete de herramientas, Sun ofrecerá un potente conjunto de componentes reutilizables para el desarrollo y explotación de aplicaciones e red, se trata de JavaServer Engine. Éste ofrece a los desarrolladores una base preconstruida que simplifica la implementación de aplicaciones en red.

La dirección en Internet en la que puede consultar sobre estos productos es **www.sun.es**

Con Host View Server de Attachmate...
ACCESO A DATOS DEL HOST MEDIANTE TECNOLOGÍA JAVA O ACTIVEX

Host View Server, desarrollado por Attachmate, líder mundial en software de acceso al host, permite el acceso al host 3270 y AS/400 utilizando tecnología Java o ActiveX.

Esta solución permite a los usuarios corporativos acceder a los datos del host mainframe a través de redes de área local, Internet o Infovía.

Las principales mejoras que presenta Host View Server, con respecto a sus anteriores versiones son los procedimientos de seguridad, la impresión y la administración.

Host View Server se apoya en unos procedimientos normalizados de seguridad (SSL 3.0) y de encriptación de datos (RSA) que aseguran la integridad de las informaciones estratégicas compartidas en Internet.

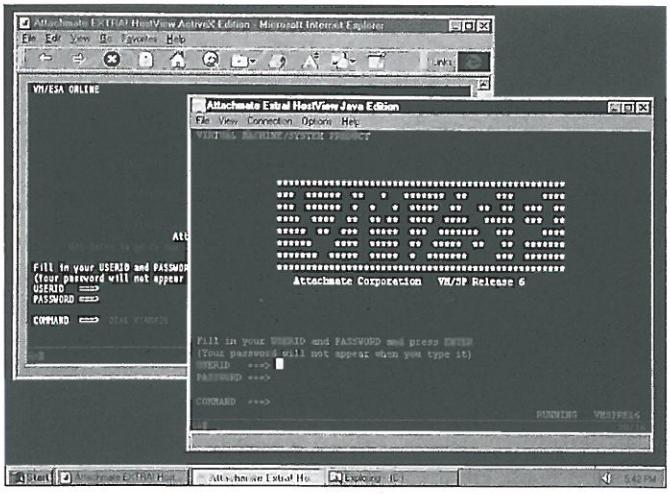
La versión 2.0 de Host View Server cumple perfectamente las normas y transferencias de impresión de ficheros en modo TN3270E y TN3287.

Host View Server es también el único producto del mercado que permite acceder a los datos del host con un simple navegador, y especialmente bajo componentes Java o ActiveX.

Además ha sido concebido como un producto Servidor para facilitar la administración centralizada de puestos clientes ligeros que acceden ocasionalmente a los datos del host.

Cualquiera que sea la solución elegida, Host View Server dispone de un "Security Proxy" que es la solución ideal para ser utilizado en las aplicaciones que requieren una seguridad absoluta de los datos.

Puede ampliar la información sobre este producto en la página de Internet del fabricante. Su dirección es **www.attachmate.com.**



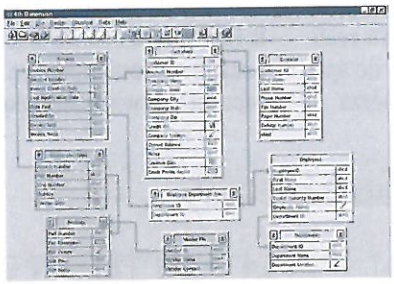
Si le gustaron otras bases de datos, 4th Dimension v6 le sorprenderá

4th Dimension es el entorno de bases de datos integrado más completo del mercado. Años de experiencia en el diseño de bases de datos han permitido satisfacer las siguientes necesidades:

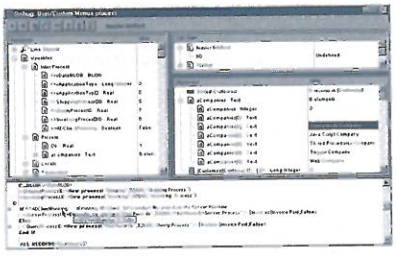
- Automatizar cualquier proceso.
- Diseñar modelos gráficos eficientes y orientados a objeto.
- Suministrar la más amplia gama de herramientas para facilitar el diseño y mantenimiento de bases de datos.
- Permitir una fácil reutilización de código.
- Permitir escalar desarrollos Cliente/Servidor, web y multiplataforma de forma transparente.

Estas son algunas de las **características** de 4th Dimension:

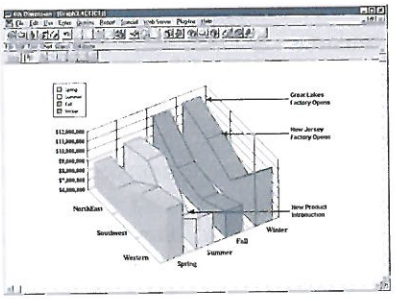
- **Motor de base de datos multiproceso, triggers, procedimientos almacenados**, presentación visual de tablas mediante diagrama de entidad /relación (ERD)



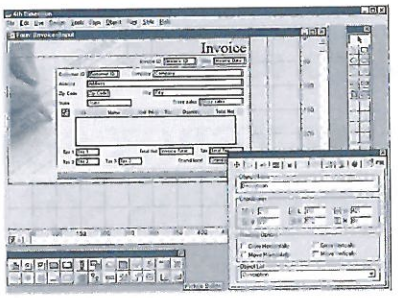
- **Debugger visual** multiproceso de última generación. **Intérprete** incluido para testeo y debugging.



- **Wizards avanzados** para diseño de formularios, queries, informes, importación, exportación, etiquetas, etc. Editor de gráficas programable 2D y 3D, soporta gráficas de barras, tartas, áreas, líneas, etc.

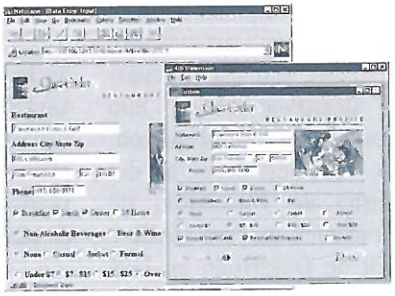


- **Editor de formularios basado en objetos.** Gestión de hojas de estilo. Redimensionamiento automático. **Docenas de objetos de interface** incluyendo check boxes, radio buttons, push buttons, áreas de scroll, picture button, botones invisibles, tab control, listas jerárquicas, termóme-tros, reglas, combo box, listas menú desplegables, menús pop-up jerárquicos, menús picture, radio pictures, listas desplegables, menús pop-up, subforms, group boxes, etc.

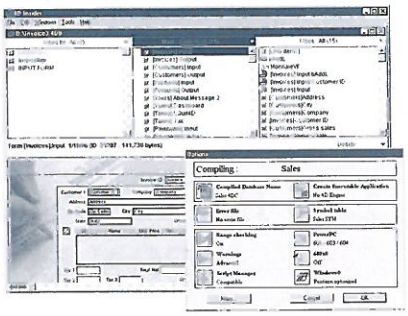


- Arquitectura de distribución de aplicaciones integrada y **soporte para proyectos con múltiples desarrolladores** con check-in/check-out automatizado.
- Arquitectura extensible mediante **plug-ins** y DLLs.

- **Un sólo código fuente** desplegable sin modificaciones de monousuario a 4D Server, la versión Cliente/Servidor de 4th Dimension.
- Soporte de **ODBC** (cliente y servidor) y conexiones nativas **SQL** a ORACLE, Sybase y MS SQL Server.
- **Servidor Web integrado** para acceso automático a formularios y gestión Cliente/Servidor mediante navegadores web.



- **Compilador a código máquina real** optimizado para diferentes procesadores. **Gestor gráfico de referencias cruzadas** y librerías de objetos.



VERSIÓN DEMO GRATUITA

4th Dimension es el núcleo de la gama de productos 4D. Visite http://www.ambitconsulting.com o llame al 93-209 41 11 para más información o para recibir una versión demo gratuita. Mencione este anuncio y reciba 4th Dimension v6 por el precio promocional de 34.900 Ptas*. (precio habitual: 79.000 Ptas.).

Balmes, 297, 4º, 2ºb. 08006 Barcelona
t. | (34) 93 209 41 11 f. | (34) 93 240 18 88
e-mail: ambit@ambitconsulting.com
http://www.ambitconsulting.com

AMBIT Consulting

(* Actualización competitiva. Oferta válida hasta el final de existencias)

Obtiene certificación como entorno de desarrollo Java

ACUERDO DE PROGRESS SOFTWARE CON SUN MICROSYSTEMS DE LINCENCIA

La empresa Progress Software, proveedor líder de herramientas de desarrollo de aplicaciones, tecnologías de bases de datos y servicios de soporte para aplicaciones empresariales, anuncia el acuerdo alcanzado con Sun Microsystems con el que obtiene la certificación del Entorno de Desarrollo de Aplicaciones en Java (JAE) y el Kit de Desarrollo en Java (JDK 1.1).

Según se confirma con este acuerdo, Progress Software integrará JAE y JDK 1.1 de Sun, en Apptivity 2.0, la herramienta de Progress para construir e implantar aplicaciones de bases de datos altamente escalables, en plataformas cruzadas de negocio.

Con estas dos tecnologías integradas en el producto será mucho más fácil para los usuarios de Apptivity 2.0, implantar aplicaciones empresariales Java en una amplia gama de plataformas.

Si desea ampliar la información sobre este u otro producto de Progress Software puede acudir a su dirección en Internet **www.progress.com** o sobre Java en **www.attachmate.com.**

Para satisfacer las necesidades del mercado

LA SOLUCIÓN RENAISSANCE DE ROSS SYSTEMS MIGRA A LA PLATAFORMA SOLARIS DE SUN

La empresa Ross Systems, suministrador líder de soluciones de software corporativo para el mercado cliente/servidor, ha anunciado la migración de su software Renaissance CS hacia la plataforma Solaris, la versión de Sun Microsystems líder en la industria del sistema operativo UNIX.

Al mismo tiempo Ross Systems ha seleccionado la plataforma SPARC/Solaris de Sun como la mejor para realizar futuros desarrollos de aplicaciones, lo que refuerza el compromiso existente con la flexible y abierta arquitectura UNIX.

Renaissance CS se basa en satisfacer las necesidades del mercado medio, compañías y divisiones de todo tipo de grandes empresas. La empresa comercializa aplicaciones de Recursos Humanos, gestión de material, gestión de transporte, distribución, etc.

Puede obtener y ampliar la información sobre esta compañía y sus productos en la dirección **www.rossinc.com.**

Como miembro estratégico de SunReady

LAS IMPRESORAS DE XEROX SERÁN COMPATIBLES CON LOS ENTORNOS SOLARIS Y SUN

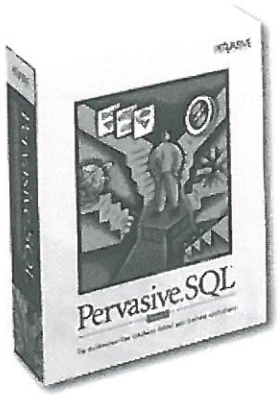
Sun ha nombrado recientemente a Xerox miembro estratégico de su nuevo programa SunReady para asegurar que las soluciones de impresión avanzada de Xerox son suministradas a los usuarios de Sun para las versiones actuales y futuras de Solaris y Java.

Como parte de este acuerdo, Xerox y Sun trabajarán conjuntamente para optimizar las soluciones de impresión del primero para los sistemas operativos de Sun y comercializarán conjuntamente estas soluciones para sus respectivos clientes a través de los canales habituales.

Equipos como las impresoras láser de red DocuPrint N32 y N24 ya están certificados como SunReady, tras haber superado las duras pruebas de compatibilidad.

Las impresoras SunReady serán compatibles con PostScript, preparadas para red y distribuidas con los servicios CentreWare de Xerox bajo SunOS/Solaris para el software de administración de impresión de las impresoras de red DocuPrint.

De cara a futuros proyectos, Sun compartirá con Xerox la información acerca de sus sistemas operativos y estrategias, además de proporcionar apoyo a la ingeniería, el producto y es soporte técnico para los desarrollos de Xerox. Puede encontrar más información en **www.xerox.es.**



Nuevo Pervasive SQL 7

El sistema
Cliente/Servidor
creado para que las **PYMES** Acierten

El sucesor de Btrieve

Pervasive SQL le ofrece lo mejor de ambos mundos:
Las mejores características de una arquitectura navegacional emparejada con la potencia de las capacidades relacionales.
Con esta estructura única, Pervasive SQL ofrece su imbatible gestión de transacciones y el máximo control en programación en adición a sus excepcionalmente rápidas consultas y manipulaciones de datos.
Pervasive SQL le ofrece lo último en operaciones confiables, sin mantenimiento, eficientes en recursos, con un rendimiento y fiabilidad de primera clase.
Pervasive SQL es la solución completa para gestionar bases de datos, con la gama más amplia de características para cubrir sus necesidades actuales y futuras. Piense si el sistema que busca o si su sistema actual le ofrece tantas ventajas como Pervasive SQL :

- ✓ Base de datos de 64 Gb.
- ✓ Motor multihilo de 32 bits.
- ✓ Operaciones continuas.
- ✓ Algoritmo de indexación hashing.
- ✓ Bloqueo a nivel de registros.
- ✓ Transacciones duraderas
- ✓ Balanceo automático de índices.
- ✓ Navegación bidireccional.
- ✓ Cache configurable.
- ✓ Integridad referencial.
- ✓ Optimización basada en costes
- ✓ ODBC
- ✓ Diccionario de datos
- ✓ Registro de archivos de Bitácora.
- ✓ Rehacer (Roll Forward).
- ✓ Expansión dinámica de ficheros.
- ✓ Utilidades.
- ✓ Manejo de licencias.

CARACTERISTICAS	Pervasive SQL 7	Su sistema de Base de Datos
Acceso Relacional y Transaccional	Si	¿?
Operaciones Libres de Mantenimiento	Si	¿?
Escalable desde Laptop a Cliente/Servidor	Si	¿?
Soporte para NetWare y Windows Nt	Si	¿?
Alto Volumen /Bajo Coste de Licencias	Si	¿?
Pequeña huella en memoria	Si	¿?
Recuperación automática	Si	¿?

Con uno de los más bajos costes de propiedad disponibles y el rendimiento y potencia requeridos para competir en el mercado actual, Pervasive SQL "es UNICO en su clase".

☐ Deseo más información ☐ Deseo asistir a una presentación

Nombre : _____
Compañía : _____
Dirección : _____
Población : _____
C.Postal y Provincia : _____
Tfno : _____ Fax : _____
Email : _____
Lenguaje programación : _____

Danysoft Internacional
Avda. España 17
28100 Alcobendas - Madrid
Tfno. (91) 6638683
Fax. (91) 6546382
Email : info@danysoft.com
www.danysoft.com

+información en www.danysoft.com

VISUAL BASIC

Creación de controles ActiveX(III)

Jordi Agost (agost@eup.udl.es)

En este artículo, vamos a continuar explicando la creación de controles ActiveX.
Ahora que ya conocemos las bases y el comportamiento de dichos controles, vamos a ampliar un poco los horizontes, agregando a los controles ActiveX características para Internet así como la posibilidad de unirlos con controles de datos.

AGREGAR A LOS CONTROLES CARACTERÍSTICAS PARA INTERNET

Cuando creamos un control ActiveX con Visual Basic 5.0, y dicho control lo colocamos dentro de un contenedor (no cualquiera, sino debe contar con funciones apropiadas para el caso, como por ejemplo MSIE), dicho control puede realizar una transferencia asíncrona de los valores de sus propiedades. Por ejemplo, si nuestro control estuviese constituido entre otros por un control **Picture**, entonces lo podríamos utilizar para pasar mapas de bits a través del mismo cambiando su propiedad *Picture*.

Los controles solicitan la transferencia asíncrona de una propiedad (en definitiva de datos) llamando al método *AsyncRead* del objeto *UserControl*. El regreso a la transferencia asíncrona de dicho método será totalmente inmediato y la transferencia a partir de ese momento se realizará siempre en un segundo plano.

La sintaxis de AsyncRead es la siguiente:

```
objeto.AsyncRead Destino, TipoAsínc [, Propiedad]
```

donde *Destino* es una cadena que especifica la ubicación de los datos, pudiendo ser una ruta de acceso o una dirección URL. Será en cada caso el *host* el encargado de establecer un tipo de comunicación u otra, según convenga. *TipoAsínc* es un entero que especifica cómo se mostrarán los datos, véase la tabla 1 para ver sus posibles valores. Por último, *Propiedad* es una cadena (opcional) que especifica el nombre de la propiedad que se va a cargar. Dicho parámetro puede ser un nombre arbitrario, ya que su única función es la de actuar como identificador de esta solicitud particular de datos, y de ser el enlace posterior para otros eventos, como vamos a ver a continuación:

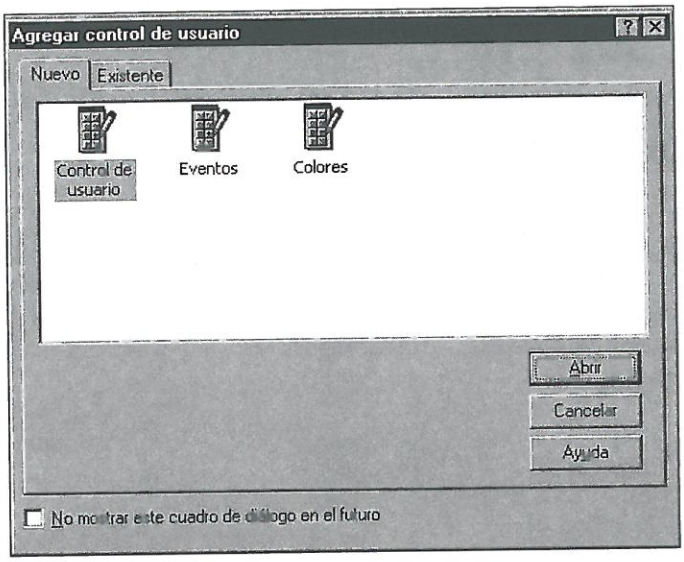
La lectura asíncrona se puede cancelar antes de que termine, mediante una llamada al método *CancelAsyncRead*, el cual tiene la siguiente sintaxis:

```
objeto.CancelAsyncRead [Propiedad]
```

donde *Propiedad* es una cadena (opcional) que especifica el nombre de la solicitud de lectura asíncrona de datos que desea cancelar. Todas las demás lecturas asíncronas continuarán normalmente, y si no se especifica en el método dicho argumento entonces se cancelará la última llamada al método *AsyncRead* que no tuviese el parámetro *Propiedad*.

Tabla 1.- Valores de la propiedad TipoAsync del método AsyncRead.

Valor	Significado
vbAsyncTypeFile	Los datos están en un archivo creado por Visual Basic.
vbAsyncTypeByteArray	Los datos se dan como una matriz de bytes que los contiene.
VbAsyncTypePicture	Se proporcionan en un objeto Picture.



Inclusión de un control de usuario.

Suponiendo que el método anterior no se haya invocado y la transmisión esté completa, el control recibirá un evento *AsyncReadComplete*, el cual consta de la siguiente sintaxis:

```
Private Sub
objeto_AsyncReadComplete(AsyncProp As
AsyncProperty)
  [código del usuario]
End Sub
```

Dicho método lleva como argumento el valor que identifica la propiedad cuyo valor se ha obtenido (ya que la transmisión ya está completa). A partir de ese momento el control tendrá acceso a los datos y podrá establecer la propiedad (sería absurdo, por ejemplo, establecer la propiedad *Picture* si no tenemos todo el mapa de bits completo).

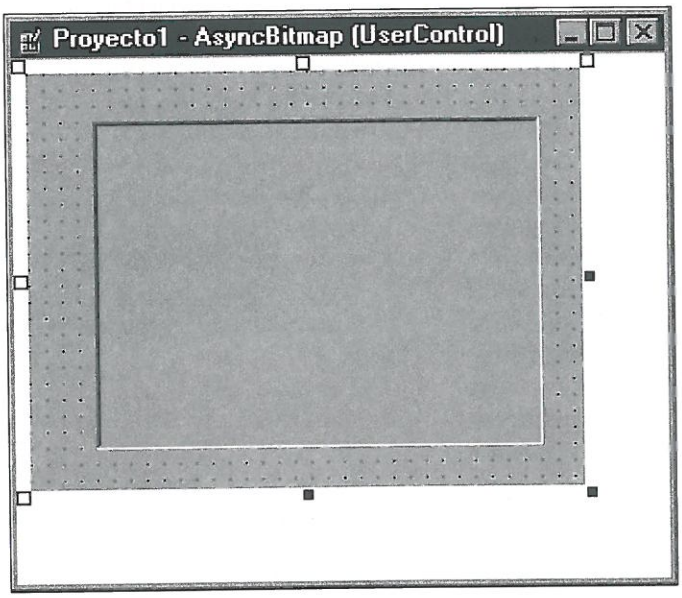
En resumen, debemos seguir cinco pasos para realizar una transferencia asíncrona que logre un buen resultado final (supongamos que se trata de un mapa de bits):

1. Situar el control ActiveX que va a iniciar la transferencia en una página Web y activar esta última.
2. Se asigna una propiedad URL (o una ruta de acceso) indicando los datos que deseamos cargar.
3. El explorador continua cargando en un segundo plano.
4. Cuando la transferencia está completa el control recibe un evento *AsyncReadComplete* y asigna el mapa de bits a la propiedad *Picture* del control.
5. El control ya puede mostrar el mapa de bits que hemos transferido.

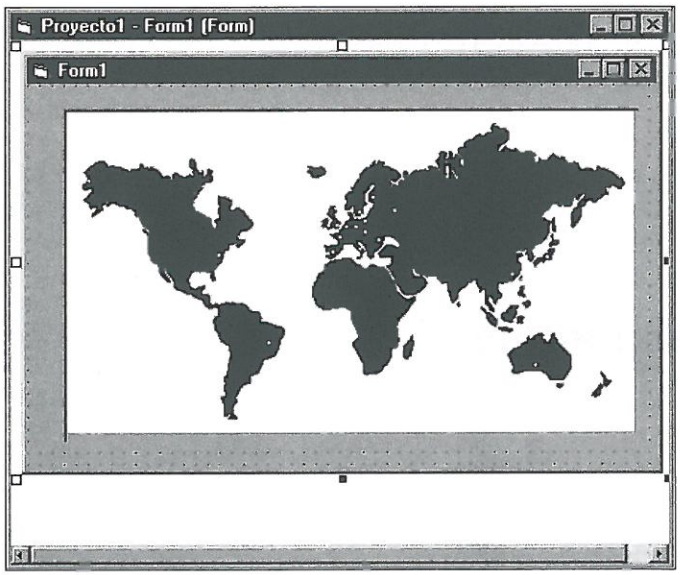
Vamos a ver ahora un ejemplo más concreto sobre lo expuesto anteriormente. Creamos un nuevo proyecto en la ventana de *Proyectos* seleccionando la opción **Nuevo Proyecto** del menú **Archivo** o mediante la combinación de teclas **Ctrl+N**. Dicho proyecto será un fichero ejecutable estándar (con extensión **EXE**, al que le añadiremos un control de usuario *UserControl*. De los tres tipos posibles: Control de usuario, Eventos o Colores, vamos a elegir el primero (ver imagen).

Ahora, dentro del control de usuario o *UserControl*, ponemos un control de tipo *PictureBox* (situado en la caja de herramientas, ver figura 1).

Posteriormente añadiremos al control *UserControl* una propiedad *Picture*, cuya misión consistirá en permitir que el control *PictureBox* que hemos añadido previamente se encargue de todo. Por tanto, a partir de este momento la propiedad *Picture* del objeto *UserControl* quedará anulada por la propiedad que hemos añadido. Si quisiéramos referirnos en



El objeto User Control con un control PictureBox en él.



Aspecto final después de que la transmisión asíncrona esté completa.

concreto a ella (a la propiedad que acabamos de anular) deberíamos referenciarla de manera explícita: *UserControl.Picture*.

La forma de añadir la *Picture* al objeto *UserControl* es la siguiente:

```
Public Property Get Picture() As Picture
  Set Picture = picBitmap.Picture
End Property

Public Property Let Picture(ByVal NewPicture As
      Picture)
  Set picBitmap.Picture = NewPicture
  PropertyChanged "Picture"
End Property

Public Property Set Picture(ByVal NewPicture As
      Picture)
  Set picBitmap.Picture = NewPicture
  PropertyChanged "Picture"
End Property
```

Como ya vimos en un artículo anterior el método *PropertyChanged* (invocado siempre al final de todos los *Property Let* o *Property Set*) sirve para notificar a Visual Basic que el valor de una propiedad ha cambiado. Recordemos que dicho paso se realiza para que VB marque las instancias que deben guardarse y éstas puedan recibir eventos *WriteProperties*. Además debemos informar al entorno de desarrollo cuándo cambia el valor de una propiedad para que pueda sincronizar los valores de la misma mostrados en las diferentes localizaciones, como por ejemplo en el cuadro de diálogo *Páginas de Propiedades*, en la ventana *Propiedades*, etc.

PropertyChanged sirve para notificar a Visual Basic que el valor de una propiedad ha cambiado

Añadimos ahora una propiedad *URL* que inicie la transferencia del mapa de bits y una vez terminada establezca dicho mapa de bits a la propiedad *Picture*.

```
Public Property Get URL() As String
  URL = strURL
End Property

Public Property Let URL(ByVal Str As String)
  strURL = Str
  If (AmbientProperties.UserMode = True And
      Str <> "") Then
    AsyncRead Str, vbAsyncTypePictrue, "URL"
  End If
  PropertyChanged "URL"
End Property
```

En este último fragmento de código (Public Property Let URL) nos aseguramos que el programa se encuentre en modo de ejecución (*AmbientProperties.UserMode* = *True*) y que la cadena URL no está vacía (*Str* <> "")

A continuación cerramos el diseñador del objeto y al formulario *Form1* le añadimos una instancia del mismo.

En ARCHIVO colocamos la ruta necesaria para realizar una transmisión local

Al código del formulario que hemos añadido (normalmente **Form1** si no lo hemos renombrado), y en la sección de declaraciones añadimos el siguiente código:

```
Option Explicit
Const ARCHIVO = "file:\f4\a.bmp"
```

En la constante ARCHIVO colocamos la ruta necesaria para realizar una transmisión local (a nuestro disco duro), indicando el camino de la imagen de bits. Es muy importante que dicha ruta empiece por el prefijo "*file:*" para que así constituya una dirección URL válida.

Y en el código del Form_Load, añadimos lo siguiente:

```
Private Sub Form_Load()
  AsyncBitmap1.URL = ARCHIVO
End Sub
```

Posteriormente añadimos en el procedimiento *AsyncReadComplete* el siguiente código:

```
Private Sub
      UserControl_AsyncReadComplete(AsyncPro
      p As AsyncProperty)
  On Error Resume Next
  Select Case AsyncProp.PropertyName
    Case "URL"
    Set Picture = AsyncProp.Value
    MsgBox "Transferencia Finalizada"
  End Select
End Sub
```

La primera línea de código *"On Error Resume Next"* se utiliza por si se diese el caso de que hubiera una causa externa a nosotros que hubiese interrumpido la emisión, de tal forma que se detectaría el error.

El objeto *AsyncProperty* se pasa al evento *AsyncReadComplete* y contiene el resultado del método *AsyncRead*. Dicho método contiene tres propiedades: *AsyncType*, *Value* y *PropertyName*.

La primera ya ha sido descrita anteriormente (ver tabla número 1). *Value* especificará su valor y *PropertyName* es la que utilizamos en el código anterior, su significado depende del contexto en el cual se encuentre: en el método *AsyncRead* especifica el nombre que se asociará con la propiedad *Value* del objeto *AsyncProperty*, y para el evento *AsyncReadComplete*, que es el que nos ocupa, especifica el nombre de la propiedad que se está leyendo actualmente. Debemos tener en cuenta que obligatoriamente debe corresponder a un nombre que fue asignado al objeto *AsyncProperty* cuando fue invocado el método *AsyncRead*.

AsyncRead especifica el nombre que se asociará con la propiedad Value del objeto AsyncProperty

Tras estos pasos, ya tenemos el gráfico dentro del control, tal y como se ve en la figura adjunta:

Una de las cosas a tener en cuenta, es que el ejemplo que hemos desarrollado hasta el momento ha sido diseñado para la transferencia a una ruta local. Tal cual lo hemos desarrollado no lo podríamos utilizar en una página web, ya que los archivos EXE estándar no pueden ser usado desde otras aplicaciones. Para que el ejemplo que hemos visto pudiese ser utilizado en una página web, deberíamos agregar nuestro archivo de origen a un proyecto de creación de un control ActiveX e inmediatamente compilarlo como un con extensión **.OCX**.

El resultado del método AsyncRead contiene tres propiedades: AsyncType, Value y PropertyName

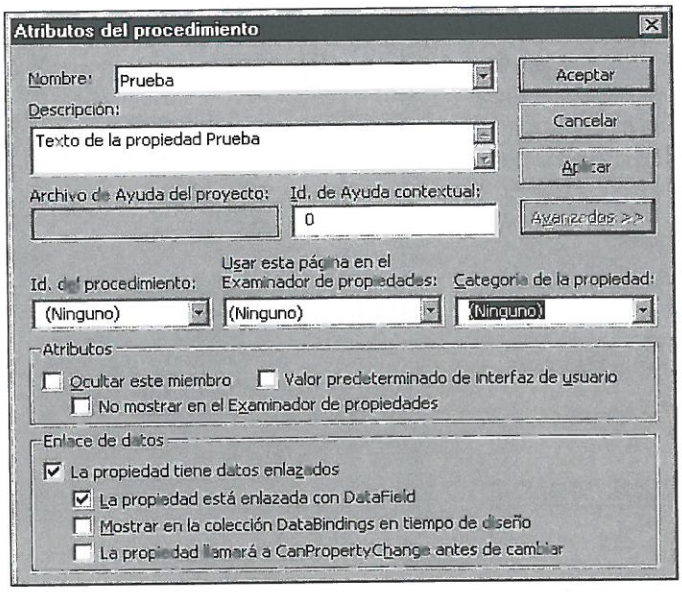
Además vamos a necesitar un poco de código adicional, por ejemplo, si queremos guardar la propiedad *Picture*, deberíamos escribir un código parecido al siguiente:

```
Private Sub UserControl_InitProperties ()
  Set Picture = Nothing
End Sub

Private Sub UserControl_ReadProperties (PropBag
      As PropertyBag)
  Set Picture = PropBag.ReadProperty
      ("Picture", Nothing)
End Sub

Private Sub UserControl_WriteProperties
      (PropBag As PropertyBag)
  PropBag.WriteProperty "Picture", Picture,
      Nothing
End Sub
```

Utilizamos *Nothing* como valor predeterminado, ya que así en el caso de que no exista una imagen, no será necesario un archivo **.FRX** suplementario.

DISEÑO DE CONTROLES PARA SU USO CON HTML

Si queremos utilizar nuestro control dentro de una página HTML, entonces debemos especificarlo con las etiquetas <OBJECT> y </OBJECT>. El navegador al procesar el código HTML creará el control y lo colocará en la página. Si dicha etiqueta OBJECT incluye algún atributo (con la sentencia <PARAM NAME>)entonces los valores de propiedad proporcionados con dicho atributo se pasarán al evento *ReadProperties* del control mediante el objeto *PropertyBag*. Si no hay ningún atributo, aparte del extensor que proporcionará propiedades tales como *Top* y *Left*, entonces puede suceder que se ejecute el evento *InitProperties* y no *ReadProperties*. Dependerá siempre de cómo se haya implementado el explorador, por lo que no es un comportamiento fiable en absoluto.



Cuadro de diálogo Atributos de Procedimiento (de la opción Prueba), con las opciones avanzadas.

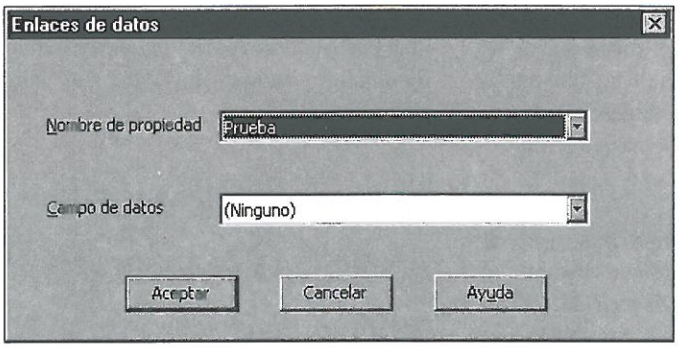
RECURSOS UTILIZADOS POR NUESTRO CONTROL

Si nuestro control utiliza muchos recursos del sistema, entonces podemos utilizar los eventos *Show* y *Hide* para anular su funcionamiento. El evento *Hide* significa que el usuario ha pasado a otra página web, con lo que si nuestro control consume muchos recursos podemos parar temporalmente su actividad para que deje el sistema más libre de recursos. Cuando el usuario vuelva a nuestra página se producirá el evento *Show* con lo que podremos reiniciar la actividad de nuestro control ActiveX.

CÓMO ENLAZAR UN CONTROL CON UN ORIGEN DE DATOS

Otro gran tema que puede ampliar de forma insospechada el campo de acción de los controles ActiveX es el hecho de poder marcar propiedades de dichos controles como enlazables. El programador podrá enlazar dichas propiedades a unos determinados campos de bases de datos, y así dotar al control de una mayor flexibilidad.

Cuando el usuario vuelva a nuestra página se producirá el evento Show reactivando control ActiveX

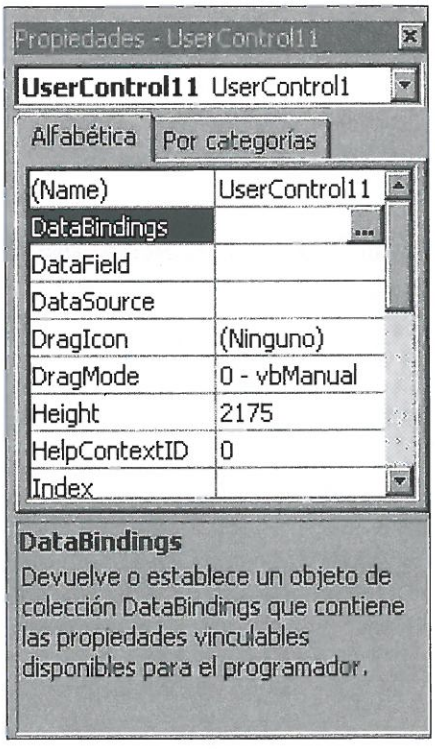


Cuadro de enlace de datos.

Para tal efecto, el proyecto anterior o en su defecto en uno nuevo, creamos una propiedad llamada *Prueba*, y entonces en el cuadro de dialogo *Atributos de Procedimiento*, escogemos las opciones *Avanzadas*, tal y como se ve en la figura 2.

En el apartado de enlace de datos, vemos que si activamos la opción en que dicha propiedad posee datos enlazados, se activan tres propiedades más.

La primera *La propiedad está enlazada con DataField* permite establecer que una propiedad de nuestro control se enlace con el campo especificado en la propiedad *DataField*. Normalmente, dicho paso se hará con el dato más importante que contenga el control.



Cuadro de UserControl.

Mostrar en la colección DataBindings en tiempo de diseño nos permitirá que la propiedad enlazable que estamos modificando, se vea en el cuadro de diálogo *Enlaces de datos*. Entonces, cuando incluyamos una instancia de nuestro control en un formulario dado y activemos la opción *DataBinding* en las propiedades (ver figura 3), aparecerá un cuadro de diálogo (ver figura 4) que mostrará todos los campos disponibles en el origen de datos, especificado este último por la propiedad *DataSource* de la instancia del control.

DataBindings ofrece VB a cualquier usuario de nuestro control y será la utilizada para enlazar propiedades adicionales

La colección *DataBindings* es lo que se llama una propiedad extendida que ofrece VB a cualquier usuario de nuestro control y será la utilizada por los programadores para enlazar propiedades adicionales (propiedades adicionales que por su naturaleza también deben ser enlazables). El total de las propiedades enlazables sólo aparecerá en tiempo de ejecución, en cambio en tiempo de diseño solamente se mostrarán las que tengan dicha opción activada.

La tercera opción, indica que el control deberá llamar a *CanPropertyChange* antes de cambiar el valor de una propiedad que pueda estar enlazada a datos. En el código deberemos chequear si *CanPropertyChanged* devuelve *True* o *False*. En el primer caso podemos establecer la propiedad, en caso contrario, nos deberemos abstener de hacerlo, ya que dicho paso puede provocar errores en algunos contenedores de controles.

De todas formas Visual Basic, siempre devuelve True (aunque en un futuro pueda no hacerlo) cuando revisamos el valor de *CanPropertyChange*, incluso si nuestro campo enlazado es de solo lectura. Lo único que sucederá es que en este último caso no actualizará el origen de datos.

Una vez creado el control, y en el caso de que un programador coloque una instancia del mismo en su formulario y deseara obtener las propiedades enlazables del mismo, deberá ejecutar el siguiente fragmento de código:

```
Dim d As DataBinding
For Each d In Form1.DataBindings
    MsgBox d.PropertyName
Next
```

donde Form1 será el nombre del formulario donde el programador ha incluido nuestro control.

De este modo los controles ActiveX pueden comunicarse con bases de datos

Supongamos ahora que el programador ha realizado dicha acción y que lo que desea es enlazar la propiedad *Prueba* con el campo *Prb* (para que este paso funcione este campo debe estar disponible en el origen de datos especificado por la propiedad extendida *DataSource*). Para ello ejecutaría el siguiente código (en tiempo de ejecución):

```
Form1.DataBindings("Prueba").DataField =
"Prb"
```

Para saber si debemos llevar a cabo el paso anterior podemos chequear, al igual que en los controles de datos, la propiedad *DataChanged* del objeto *DataBinding*.

En este artículo hemos visto cómo ampliar nuestros controles ActiveX de forma que puedan adquirir capacidades para comunicarse con bases de datos e Internet. Después de estos artículos sólo nos queda tratar más a fondo el tema de las propiedades, y algunos aspectos avanzados del objeto UserControl.

Concurso de Programación

Como recordaras teníamos preparado un concurso de programas de ajedrez, pues bien .... el mismo queda convocado para los dias 12 y 13 de Septiembre, siendo la fecha límite de recepción de programas el dia 9 del mismo.

Para más información e inscripciones, puedes ponerte en contacto con nosotros en el teléfono 91 559 82 86 y pregunta por la Srta. SANDRA.

SÓLO PROGRAMADORES

AWS AWS

Programación del acelerador gráfico Intel740

Jorge Figueroa (erde@arrakis.es)

Actualmente el mercado de las aceleradoras 2D/3D ha tenido un aumento casi geométrico y todos los juegos y programas multimedia más modernos hacen uso de ellas, ya sea por medio de DirectX o de OpenGL.
En un momento de tanta competencia, Intel irrumpe con fuerza en este mercado con su nuevo chip i740.

Todos los juegos y programas multimedia en general del momento hacen uso de las nuevas tarjetas aceleradoras 2D/3D mediante el uso de las famosas librerías estándar de Windows *DirectX* u *OpenGL*. Gracias a esta tecnología se consiguen generar imágenes 2D/3D de alta calidad y con una elevada tasa de refresco incluyendo toda clase de efectos tales como transparencias, niebla, texturizados, antialiasings, etc.

El chip i740 está diseñado para funcionar en sistemas AGP

Todas estas placas 2D/3D aceleradoras son relativamente baratas y mucha gente empieza a incorporarlas actualmente en sus equipos. Intel, ha visto que este mercado ofrece cada vez más dividendos y finalmente se ha decidido a crear el primer chip acelerador 2D/3D de bajo coste denominado i740.

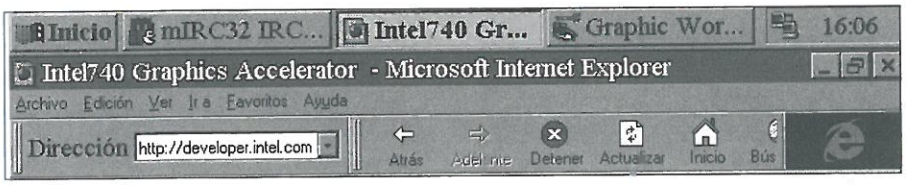
Este acelerador no ha sido creado con la intención de que sea un competidor más entre los muchos que ya hay actualmente en el mercado, sino que se ha diseñado con la idea de que se convierta en un componente estándar. Es decir, el objetivo de Intel consiste en la instalación de serie en las placas base de nueva generación que aparezcan en el mercado en un futuro inmediato, como respuesta al aumento de recursos que requieren la mayoría de las aplicaciones estándar actuales.

BUS AGP

Como era de esperar siendo una nueva tecnología gráfica, este nuevo chip acelerador 2D/3D ha sido diseñado para ser instalado en placas base basadas en la arquitectura de bus AGP (*Accelerated Graphics Port*: Puerto de Gráficos Acelerado), diseñada también por Intel para que los rápidos procesadores Pentium II actuales puedan transmitir información gráfica a gran velocidad por un bus exclusivo para ello. De esta forma, se consiguen unos ratios de transferencia de datos de hasta 533 Mb/s, junto con la liberación de los buses PCI estándar para otras operaciones de datos.

PROGRAMAR EL INTEL740

Este nuevo chip, está diseñado para poder funcionar tanto con las librerías estándar de Windows *DirectX y OpenGL*, por lo que su compatibilidad con las aplicaciones gráficas más modernas será inmediata.



En la página Web de Intel se puede encontrar toda la información referente a este nuevo chip.

CARACTERÍSTICAS TÉCNICAS DEL CHIP

Para que el lector pueda hacerse una idea de la potencia general de este chip y para que pueda compararlo con otros de los que pueda obtener información, se pueden dar algunos datos orientativos típicos de velocidad y funcionalidad: 0.85 Megatriángulos por segundo, entre 425 y 500 K de triángulos por segundo con todas las características 3D activadas, entre 45 y 55 Megapixels por segundo con todas las características activadas, uso del puerto doble AGP, ejecución en paralelo, salida de vídeo programable, soporte para captura de vídeo (16 o 8 bits), Reproducción de vídeo en pantalla completa, soporte de Intercast y VBI, capacidad de uso de MPEG II DVD. Por último decir que está optimizado para funcionar con el chipset 440LX de Intel.

A continuación se van a detallar todas las funciones 3D, 2D y otras que puede realizar el chip:

FUNCIONES 3D DEL CHIP

En el apartado de las funciones 3D, que es el apartado que más suelen apreciar todos los usuarios domésticos debido principalmente a que los videojuegos más populares y que más atractivo tienen son los simuladores 3D, se puede decir que el chip posee todas las capacidades típicas de esta clase de chips. Se trata de las siguientes:

ATRIBUTOS DE COLOR EN SUPERFICIE

Dentro del apartado de efectos típicos que se suelen aplicar a los objetos generados en escenas 3D, tenemos que el chip Intel740 soporta todos los siguientes:

EFECTO FOGGIN

El efecto *foggin* consiste en poder añadir el efecto de densidad a la atmósfera de la imagen que estemos generando, dando la sensación que los objetos se ven más *nublados* a mayor distancia del observador.

La fórmula que usa internamente este algoritmo es la siguiente: { $C = f * Cp + (1 - f) * Cf$ }, donde *f* es el color del polígono y *Cf* el color del *fog* (niiebla).

Ejemplo con DirectX:

Para activar este efecto, solamente tenemos que poner la bandera *D3DRENDERSTATE_FOGENABLE* a **TRUE** y después se debe especirficar el color *fog* usando valores RGB. Por otro lado, la densidad del efecto *fog* se debe indicar mediante el componente *alpha* del valor especular de un vértice usando el tipo de dato **pLVertex** en *D3DLVERTEX*.

Por último sólo decir que este efecto puede regularse entre los valores 0 y 255, siendo 0 el valor máximo aplicable (densidad *fog* máxima) y 255 el mínimo (equivalente a desactivado).

El código de todo lo dicho quedaría de esta forma:

```
SetRenderState(D3DRENDER_STATE_FOGENA-
    BLE, TRUE);
SetRenderState(D3DRENDER_STATE, FOGCO-
    LOR, <D3DCOLOR>);
D3DLVERTEX pLVertex;
pLVertex.specular = RGBA_MAKE(sred, sgreen,
    sblue, FOG_DENSITY);
```

Ejemplo con OpenGl:

El código necesario para que las librerías *OpenGL* activen el efecto anterior y para que usen la aceleración por hardware con el chip Intel 740 se debe indicar como sigue:

```
glEnable(GL_GOF) {...};
glFogi(GL_FOG_MODE, GL_LINEAR);
GLfloatfogColor[4] = {0.5, 0.5, 0.5, 1.0};
glFogfv(GL_FOG_COLOR, fogColor);
glFogf(GL_FOG_DENSITY, <0.5>);
```

Como detalle decir que en *OpenGL* el nivel de densidad se indica con un valor de coma flotante entre 0.0 y 1.0 y no con un valor byte entero.

LUZ ESPECULAR

Otro de los efectos mas usados en la generación de imagen 3D y que soporta el chip, es el uso de luces especulares (*Specular highlighting*), pudiéndose incluso elegir el tipo (atributos) de luz que hará el efecto.

Posee aceleración tanto 2D como 3D

Ejemplo con DirectX:

A continuación se muestran unas líneas de código que atribuyen una luz

especular roja en un vértice y que activa este efecto:

```
D3DLVERTEX pLVertex;
pLVertex.specular= RGBA_MAKE( 255, 0, 0,
    FOG_DENSITY);
SetRenderState( D3DRENDERSTATE_SPECU-
    LARENABLE, TRUE);
```

Ejemplo con OpenGL:

```
Glfloat mat_specular[]= {1.0, 1.0, 1.0, 1.0}
glMaterialfv( GL_FRONT, GL_SPECULAR,
    mat_specular);
```

ALPHA BLENDING

Con este efecto se puede dar a los objetos la propiedad de transparencia, pudiendo hacer que cada elemento posea un grado mayor o menor de transparencia (por ejemplo, para hacer una bola de cristal, un vaso, etc.).

El chip permite usar atributos de color como Fog, luz especular...

Ejemplo con DirectX:

Activar este efecto es sólo cuestión de poner a TRUE una bandera y dar unos valores a dos variables:

```
SetRenderState( D3DRENDERSTATE_ALPHA-
    BLENDENABLE, TRUE);
SetRenderState( D3DRENDERSTATE_SRC-
    BLEND, <D3DLEND FLAG>);
SetRenderState( D3DRENDERSTATE_DEST-
    BLEND, <D3DBLEND FLAG>);
```

El parámetro *D3DBLEND FLAG* puede ser: ZERO, ONE, SRCCOLOR, INVSRCCOLOR, DESTCOLOR, INVDESTCOLOR, BOTHSRCALPHA o BOTHINVSRCALPHA.

Ejemplo con OpenGL:

```
glEnable( GL_BLEND);
glBlendFunc(<SOURCE_FLAG>, <DESTINA-
    TION_FLAG>);
```

ALPHA TESTING

El chip i740 soporta el uso de funciones de testeo de Alpha Blend. Con ello, el usuario puede controlar como se aplica el efecto a los objetos en la escena.

Ejemplo con DirectX:

```
SetRenderState(D3DRENDERSTATE_ALPHA-
    BLENDENABLE, TRUE);
SetRenderState(D3DRENDERSTATE_ALPHA-
    TESTENABLE, TRUE);
SetRenderState(D3DRENDERSTATE_ALPHA-
    FUNC, <D3DCMPFUNC>);
SetRenderState(D3DRENDERSTATE_ALPHA-
    REF, <ALPHA REF>);
```

El parámetro *<D3DCMPFUNC>* puede ser uno de estos valores: D3DCMP_NEVER, D3DCMP_LESS,D3DCMP_EQUAL, D3DCMP_LESSEQUAL, D3DCMP_GREATER, D3DCMP_NOTEQUAL, D3DCMP_GREATEREQUAL o D3DCMP_ALWAYS.

Por otro lado, el parámetro *<ALPHA REF>* debe ser un número que especifique un valor de referencia Alpha para los pixels que se testeen cuando este modo se encuentre activado.

Ejemplo con OpenGL:

Con OpenGL, su activación es más sencilla, y el código sería el siguiente:

```
glEnable(GL_ALPHA_TEST);
glAlphaFunc(<GLFUNC>, <GLREF>);
```

En el ejemplo, <GLFUNC> puede ser GL_NEVER, GL_ALWAYS, GL_LESS, GL_EQUAL, GLGEQUAL, GL_GREATER o GL_NOTEQUAL y el valor *<GLREF>* debe estar comprendido entre 0 y 1 (como siempre en *OpenGL*).

DIFUMINADO DE COLOR

Este efecto (en inglés *Color Dithering*), consiste en que cuando un patrón de pixels (una textura) que posee varios colores bastante juntos se coloca a una distancia del observador en imagen como para que no quepan todos ellos representados gráficamente (o sea, que haya varios pixels del patrón ocupando un pixel de pantalla), este mecanismo calcula el nuevo color resultante que se vería a partir de su mezcla.

Este efecto funciona cogiendo el *color de entrada*, el valor *alpha*, el valor *fog* y los convierte de un valor de 8 bits a otro de 5 o 6. El efecto de difuminado simula 256 niveles de color mediante el uso de una tabla de 32 o 64 pixels.

Ejemplo con DirectX:

Solamente hay que incluir la línea:

```
SetRenderState(D3DRENDERSTATE_DITHERE-
    NABLE, TRUE);
```

Ejemplo con OpenGL:

```
Gl_Enable(GL_DITHER);
```

SOMBREADO

El efecto de sombreado determina como deberán ser interpolados los colores de cada uno de los pixels de la superficie de cada triángulo (polígono). Cada atributo de color: *alpha*, efecto *fog*, luz especular y atributos de color se sombrean de forma individual.

El chip i740 soporta dos métodos de sombreado: *Flat Shading* y *Gouraud Shading*.

El método de sombreado *Flat* hace que cada polígono de un objeto se dibuje con sólo un solo, es decir, se rellena con color sólido.

Por otro lado, la técnica Gouraud hace que dentro de cada polígono se hagan progresiones de color entre cada uno de los vértices adyacentes, de forma que no se podrán diferenciar los límites de cada polígono ya que no hay saltos bruscos de color entre ellos.

USO DE TEXTURAS

El chip Intel740 tiene capacidad para poder manipular una cantidad ilimitada de texturas (mapas de bits). La razón de ello es que al ser un acelerador que se instala en sistemas que usan la tecnología AGP, no usa memoria propia para almacenar las texturas como hacen todas las placas aceleradoras actuales, si no que usa como memoria local de vídeo la misma memoria RAM. Como resultado el sistema puede usar tanta memoria RAM de vídeo como necesite, que es lo mismo que decir que no se tiene límites (más que los de arquitectura del chip 32 o 64 bits de espacio de memoria direccionable y el de la memoria instalada en el equipo).

En texturas, soporta el MipMapping y la interpolación bilinear

Para aquellos que no lo sepan, decir que la tecnología de bus AGP permite que el chip gráfico (en este caso el Intel740) pueda acceder a las texturas sin la necesidad de la intervención de la CPU. Este sistema de buses de transferencia paralelos, permite que el sistema pueda alcanzar un ancho de banda de hasta 1.3 Gbytes por segundo.

Además de lo dicho, el chip posee varias funciones para la manipulación de las texturas y es compatible con bastantes formatos de textura. A continuación se detallan estos aspectos:

FORMATOS DE TEXTURA SOPORTADOS

El chip de Intel soporta el uso de texturas con más de 16 bits de color en varios formatos, que pueden catalogarse en tres categorías: **ARGB, AYUV y YUB.**

La única limitación que posee el Intel740 es que la dimensión de las textura debe tener un valor que sea potencia de dos.

Es compatible con 11 formatos de textura

En la tabla 1 se muestra la lista completa de los formatos soportados por el Intel740 compatibles con *DirectX* y *OpenGL*. Así mismo, en la tabla 2 se da un ejemplo completo de cómo crear una textura en memoria convencional desde *DirectX 5*.

MEZCLADO DE TEXTURAS

Esta es una de las funciones que puede efectuar el chip con las texturas y consiste en la posibilidad de mezclar los colores de la propia textura con los del vértice asociado.

Tabla 1.- Formatos de textura soportados por el i740

1-Compatibles con DirectX:
-1555 ARGB.
-0565 ARGB (por defecto para modo paletizado).
-4444 ARGB (por defecto para alpha).
-422YUV.
-422YUV.
-0555AYUV (Sistema con compresión).
-1544AYUV (Sistema con compresión).
-Paletizado de 1, 2, 4 y 8 bits.

2-Compatibles con OpenGL:
-RGB (0555ARGB).
-RGBA4 (4444ARGB).
-RGB5_A1 (1555ARGB).

TRANSPARENCIAS EN TEXTURAS

El chip dispone de dos efectos aplicables a las texturas llamados *Texture Map Chroma Keying* y *Color Keying*. Ambas técnicas sirven para lo mismo, permitiendo seleccionar un color de la textura para que sea invisible (sea transparente, no presente), de forma que al dibujar en pantalla las partes transparentes dejan ver lo que hay detrás de la propia textura.

La única diferencia que hay entre ambas técnicas, es que la primera se usa para texturas paletizadas, mientras que la segunda se usa en texturas sin paleta, donde en consecuencia, se debe indicar el color transparente mediante un valor de 16 bits.

FILTRADO DE TEXTURAS

Este efecto es muy apreciado y da mucha más calidad a las imágenes 3D. Simplemente consiste en que el hardware

Tabla 2.- Ejemplo de código de cómo crear una textura en formato 4444 ARGB en memoria convencional del sistema mediante DirectX

```
DDPIXELFORMAT ddpf;
DDSURFACEDESC ddsd;
ddpf.dwsize = sizeof(ddpf);
ddpf.dwsize = sizeof(ddsf);
ddpf.dwRGBBitCount= 16;
ddpf.dwRBitMask = 0x0F00h;
ddpf.dwGBitMask = 0x0F00h;
ddpf.dwBBitMask = 0x000Fh;
ddpf.dwRGBAlphaBitMask = 0xF000;
ddpf.dwFlags = DDPF_RGB;
ddsd.dwFlags = DDSD_PIXELFORMAT;

IDIRECTDRAW*lpdd;
IDIRECTDRAWSURFACE*lpTextureSurface;
HRESULT ddrval;
ddsd.dwsize = sizeof(ddsd);
ddsd.dwHeigh = 128;
ddsd.dwWidth = 128;
ddsd.wFlags = DDSD_CAPS | DDSD_HEIGHT | DDSD_WIDTH;
ddsd.ddsCaps = DDSCAPS_TEXTURE | DDSCAPS_ALLOCONLOAD | DDSCAPS_VIDE-
    OMEMORY |
DDSCAPS_NONLOCALVIDMEM;
ddrval = lpdd->CreateSurface(&ddsd, &lpTextureSurface, NULL);

IDIRECTDRAWPALETTE *lpDDPal;
lpDDPal = DDLoadPalette(lpDD, "MiTextura.bmp");
ddrval = lpTextureSurface->SetPalette(lpDDPal);
ddrval = DDReLoadBitmap(lpTextureSurface, "MiTextura.bmp");

D3DTEXTUREHANDLE HTex;
lpTextureSurface->GetHandle(lpD3Ddevice, &HTex);
SetRenderState(D3DRENDERSTATE_TEXTUREHANDLE, &Htex);
```

usa un sistema de cálculo mediante el cual se consigue que las texturas que se dibujan en las escenas 3D queden mejor visualmente sea cual sea la distancia (escala) a la que se encuentren dibujadas.

Para hacer esto, el filtrado de texturas hace que el chip calcule el color del pixel de la textura que se verá en pantalla según la influencia de varios factores, tales como es el de la distancia entre el punto y el observador y el tamaño de la textura en relación al tamaño con el que aparece en pantalla; de la manera en que lo veremos, según el procedimiento.

MÉTODOS DE FILTRADO

- *Sistema Nearest (de proximidad):* Este sistema, también llamado *filtrado de puntos*, simplemente usa un algoritmo que pone en cada punto de pantalla el pixel que más próximo queda en la textura.

  Este sistema tiene el inconveniente que cuando la textura en pantalla se amplia mucho se ven grandes cuadros monocolor (pixels gigantes), lo que da una apariencia poco realista al conjunto.

- *Sistema Lineal:* Este modo, conocido también como *filtrado bilineal*, consiste en que para dibujar cada punto el chip calcula dos coordenadas cercanas y coloca el pixel con el color intermedio resultante de ambos (la media). El resultado de este sistema es que las texturas aparecen más suavizadas en pantalla, sin tanta pixelación en las ampliaciones.

Soporta resoluciones desde los 640x480 hasta los 1280x1024 este sistema tiene el inconveniente de dar una apariencia poco realista

Ejemplo con DirectX:

```
SetRenderState(D3DRENDERSTATE_TEXTURE-
    MIN, <D3DTEXTUREFILTER>);
SetRenderState(D3DRENDERSTATE_TEXTURE-
    MAG, <D3DTEXTUREFILTER>);
```

La variable *D3DTEXTUREFILTER* puede poseer los valores: D3DFILTER_NEAREST,D3DFILTER_LINEAR,D3DFILTER_MIPNEAREST, D3DFILTER_MIPLINEAR,D3DFILTER_LINEARMIPNEAREST y D3DFILTER_LINEARMIPLINEAR.

Ejemplo con Open GL:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTU-
    RE_MIN_FILTER, <FILTER_MODE>);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTU-
    RE_MAG_FILTER, <FILTER_MODE>);
```

En este caso, la variable *FILTER_MODE* puede poseer estos valores: GL_NEAREST, GL_LINEAR, GL_MIPMAP_NEAREST, GL_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_NEAREST y por último GL_LINEAR_MIPMAP_LINEAR.

LA TÉCNICA MIPMAPPING

Las texturas pueden verse desde muchas distancias en una imagen 3D, y una consecuencia de ello es que hay ocasiones en las que algunas texturas deben aparecer en pantalla más pequeñas de lo que realmente son (se ven con menos detalle del que realmente tienen). Cuando pasa esto, si no se aplica ningún algoritmo especial, el resultado es que la imagen se dibuja como si tuviese un efecto de interferencias o nieve (que se nota más todavía si la imagen está en movimiento como puede ser un simulador).

Para evitar este efecto visual precisamente se usa la técnica *Mipmapping*, que consiste en que el sistema posee guardadas en memoria varias copias de una misma textura cada una de ellas a una escala menor (cada una la mitad a la anterior aproximadamente). Con esto, el chip puede elegir según la distancia a la que se encuentra el objeto la textura con el tamaño que se adapte mejor a la escala de dibujado en pantalla. De esta forma se consigue que se vea mejor la textura desde todas las distancias.

LA TÉCNICA ANTIALIASING

Este algoritmo (cuyo nombre se traduciría como *antidentado*), consiste en hacer que todos los bordes externos de los triángulos que forman un objeto aparezcan en imagen sin que se vea la cuadriculación de los pixels, lo cual se nota mucho por ejemplo si dibujamos una figura negra con formas irregulares en un fondo de color blanco.

El algoritmo consiste simplemente en coger los pixels más externos y el primero que haya en contacto con el borde del fondo, buscar el color medio entre ambos y dibujar dicho color entre ambos.

Tabla 3.- Resoluciones soportadas por el chip

-Modo 640*480 en modo 256 colores con paleta.
-Modo 640*480 en modo 65536 colores (16 bits)
-Modo 640*480 en modo 16,7 millones de colores (24 bits).

-Modo 800*600 en modo 16/4.
-Modo 800*600 en modo 256 colores con paleta.
-Modo 800*600 en modo 65536 colores (16 bits)
-Modo 800*600 en modo 16,7 millones de colores (24 bits)

-Modo 1024*768 en modo 256 colores con paleta.
-Modo 1024*768 en modo 65536 colores (16 bits)
-Modo 1024*768 en modo 16,7 millones de colores (24 bits)

-Modo 1280*1024 en modo 256 colores con paleta.
-Modo 1280*1024 en modo 65536 colores (16 bits).

Con el método explicado, se consigue que desaparezca en gran medida el dentado que poseen los dibujos a causa de que las imágenes estén hechas mediante pixels (pequeños cuadrados), dando un aspecto difuminado en los bordes externos, ofreciendo la sensación de una imagen con más resolución de la que tiene realmente.

El chip puede manejar una cantidad infinita de texturas

Además de lo anterior, para que este efecto funcione es necesario que se ordenen todos los triángulos de la escena de mayor a menor distancia y que se empiece por dibujar primero los más lejanos.

Ejemplo con DirectX:

```
SetRenderState(D3DRENDERSTATE_ANTIA-
    LIAS, SORTDEPENDENT);
```

Ejemplo con OpenGL:

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,
    GL_ONE_MINUS_SRC_ALPHA);
glEnable(GL_POLYGON_SMOOTH);
```

ELIMINACIÓN DE CARAS OCULTAS

La técnica a la que estamos haciendo referencia es también muy usada y consiste en que el propio chip se encarge de calcular cuáles son los polígonos visibles para el observador en la imagen antes de dibujarlos en pantalla (es decir, aquellos que deben aparecer en imagen) para ahorrar de esta forma que dibuje polígonos innecesarios, con el consecuente ahorro de tiempo para el usuario y cálculos para el sistema que estamos utilizando.

Ejemplo con DirectX:

```
SetRenderState(D3DRENDERSTATE_CULLMO-
    DE, <MODE>);
```

MODE puede ser uno de estos valores que citamos a continuación: D3DCULL_CCW y D3DCULL_CW.

Ejemplo con OpenGL:

```
glEnable(GL_CULL_FACE);
glCullFace(<MODE>);
```

MODE puede ser: GL_FRONT, GL_BACK, o GL_FRONT_AND_BACK.

FUNCIONES DE ACELERACIÓN 2D DEL CHIP

El chip i740, como ya se ha dicho al principio del artículo, también posee aceleración para las 2D, lo que resulta muy útil para un entorno gráfico 2D como es Windows.

- *Engine*: El sistema posee un *engine* denominado genéricamente *BitBLT* (del inglés **Bit** Data **BL**ock **T**ransfer) que permite funciones hardware como las de copiar bloques de memoria de una zona de la pantalla gráfica a otra.

- *Escalado de bloques:* También ofrece la función que encargada de copiar bloques con el añadido de poder escalar el tamaño con el que se copiarán los bloques tanto en la dimensión X como en la Y por medio de un algoritmo de replicación/interpolación de pixels.

- *Conversión de formatos:* Además de las características mencionadas, los bloques que se copian pueden tener formatos de color diferente, pudiendo así por ejemplo copiar un bloque gráfico con 16 bits de color en otro de 24.

El sistema posee un engine denominado genéricamente BitBLT que permite funciones hardware como las de copiar bloques de memoria

- *Cursor por hardware:* Esta característica está presente en casi todas las aceleradoras actuales y en este caso el chip posee 4 Kbytes para el almacenado de cursores a la vez que soporta seis formatos diferentes.

OTRAS CARACTERÍSTICAS

Además de todo lo explicado en este artículo, el chip posee otras muchas capacidades que no se van a mencionar a continuación por salirse un poco del tema tratado en el artículo, como pueden ser la captura de vídeo, reproducción en formatos MPEG II, compatibilidad con los algoritmos usados para DVD, y otras capacidades propias del chip citado.

EUROPRIX 98
MultiMediaArt
Concurso europeo de creatividad
La Comisión Europea DGXIII y el Ministerio Federal Austriaco de asuntos económicos, invita a todos los producentes europeos de Multimedia al primer y único concurso de las mejores creaciones de Multimedia.
El concurso está organizado con motivo de la Presidencia de la República de Austria de la UE y abierto para todos los producentes miembros de la UE, los pertenecientes a la EFTA, los 10 países de Europa central y del este aspirantes a la UE, incluyendo Chipre.
El EuroPrix MultiMediaArt 98 convoca o reune productos Multimediales comerciales, bajo la idea : "Multimedia con un fin".
Fecha límite de entrega: Junio 30 de 1998
EuroPrix Secretariat:
c/o Techno-Z FH Forschung & Entwicklung GmbH.
A-5020 Salzburg, J.-Haringer-Str. 5,
Tel: +43-662-454888-714, Fax: +43-662-45 61 74,
e-mail: secretariat@europrix.org
Para más informaciones, le invitamos a consultar la página Web del EuroPrix:
www.europrix.org
www.europrix.org
Das Wirtschaftsministerium
midasnet

Programación de DirectX con Delphi (VI)

Constantino Sánchez Ballesteros (constantino@redestb.es)

QUÉ ES DIRECTPLAY

El API *DirectPlay* es una interfaz que simplifica el acceso de las aplicaciones a los servicios de comunicación. No sólo permite a las aplicaciones una forma de comunicarse con otras, independientemente del protocolo utilizado o servicio, sino que otorga independencia para la creación de servidores de juegos o cualquier implementación que se nos ocurra.

Con DirectPlay lograremos la interactividad entre varios usuarios conectados a una determinada red

Las aplicaciones (especialmente juegos) son más competentes si pueden ser disfrutadas entre jugadores reales que tienen opciones de conectividad desde cualquier lugar del mundo. Los programadores, siempre han tenido que ingeniárselas para permitir diferentes soluciones de conectividad entre equipos. Para paliar esto, *DirectPlay* tiene una estructura bien definida para las capacidades de comunicación más generalizadas. Mediante este API, no debemos preocuparnos de los detalles internos impuestos en los protocolos de transmisión de datos ni implementaciones complejas de conectividad; tan sólo hay que ceñirse en la creación de la aplicación y usar *DirectPlay* como si fuera una simple librería de funciones.

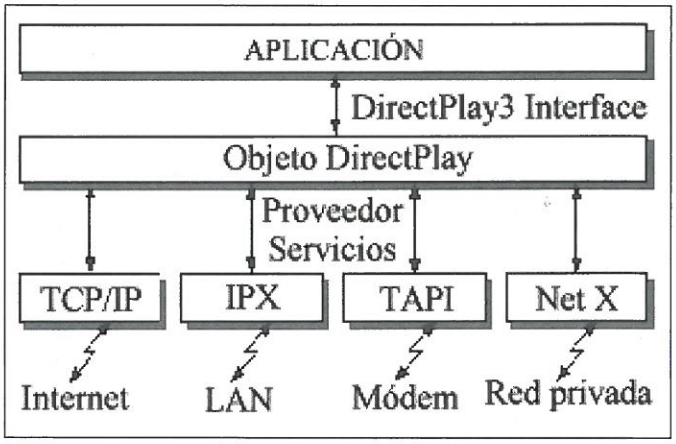
DESARROLLO DE SOFTWARE EN RED

El concepto fundamental concerniente a la creación de aplicaciones en red es cómo propagar la información a todos los ordenadores que estén conectados en una misma sesión. El estado de la sesión define lo que el usuario ve en los ordenadores conectados con él y qué acciones puede realizar. Generalmente, dos cosas camuflan el estado de la sesión:

1. El entorno.
2. Los usuarios individuales o jugadores.

El entorno consiste en los objetos que los jugadores pueden manipular o interactuar con ellos. Esto puede ser un mapa de un castillo, un circuito de carreras o un simple documento. También

En este número vamos a centrarnos en uno de los API's DirectX más importantes en la programación de juegos, debido a sus posibilidades interactivas multiusuario. Me refiero a DirectPlay. Puesto que para empezar a programar hay que saber antes los aspectos de lo que se programa (valga la redundancia).



Esquema de aplicación.

puede incluir jugadores controlados por el ordenador. En conjunción con el entorno, cada jugador tiene un estado indicando sus propiedades actuales. Éstas pueden incluir posición, velocidad, energía, armamento, etc.

Cuando un jugador se une a la sesión creada, el estado actual de la misma debe ser actualizado, incluyendo el entorno y el estado actual de los otros jugadores. Las acciones creadas por los usuarios o la progresión natural de un juego siempre cambia el estado de una sesión. Cuando este hecho ocurre, el cambio debe ser propagado a los otros ordenadores conectados a la sesión a través de mensajes. Hay muchas técnicas para actualizar el estado de la sesión, pero todas ellas dependen de los mensajes enviados entre los ordenadores.

Cuando dos jugadores realizan una acción sobre un mismo objeto (por ejemplo, el uso de un arma), o el uno sobre el otro, pueden ocurrir conflictos. Esto es, cuando dos jugadores prueban a hacer algo donde solamente uno de ellos puede efectuarlo. En estos casos, es necesaria una rutina para la resolución de conflictos. Hay varias formas de resolver este tipo de problemas:

- que los dos jugadores puedan comunicarse y decidir quien gana
- que un jugador neutral arbitre todos los conflictos

Las acciones del cambio de estado de jugadores en un juego y sus correspondientes conflictos, deben ser resueltos antes de que el estado del juego pueda ser actualizado. Hay dos formas fundamentales para mantener el estado en una sesión:

1. *Peer-to-Peer*: en este tipo de sesiones, todos los ordenadores tienen información sobre el estado completo del entorno y los jugadores involucrados. Cualquier cambio del estado ocurrido en un ordenador debe ser propagado a todos los demás. Ningún ordenador estará realmente al cargo de la sesión.

2. *Cliente/servidor*: en las sesiones de este tipo, un ordenador es designado como servidor, y mantiene un control total sobre el estado del juego resolviendo cualquier conflicto que ocurra en un momento dado. Los demás ordenadores de la sesión son clientes, y suelen actualizar alguna porción del estado del servidor. Cuando necesitan cambiar el estado, se comunican con el servidor, y éste a su vez, propaga el cambio a los otros clientes cuando se requiera.

Con *DirectPlay*, podemos crear y manipular tanto aplicaciones *Peer-to-Peer* como *Cliente/Servidor*. Además tiene todas las herramientas necesarias para crear una aplicación en red utilizando servicios para el manejo de jugadores, envío recíproco de mensajes y propagación automática del estado de los ordenadores.

FUNCIONAMIENTO INTERNO DEL API

Este API puede definirse como un sistema de objeto distribuido al cual podemos acceder. *DirectPlay* define la funcionalidad de las redes abstractas para utilizarla en nuestra aplicación, con la condición de que los equipos tengan acceso a una red. En los casos en que un método no esté soportado por la configuración de red instalada, *DirectPlay* contiene el código necesario para emularlo.

La arquitectura del proveedor de servicios de *DirectPlay* aísla a la aplicación de la red que se esté utilizando. Nuestra aplicación puede preguntar a *DirectPlay* por las capacidades específicas de la red instalada (por ejemplo, latencia y ancho de banda) y ajustar las comunicaciones de forma acorde a los parámetros recibidos.

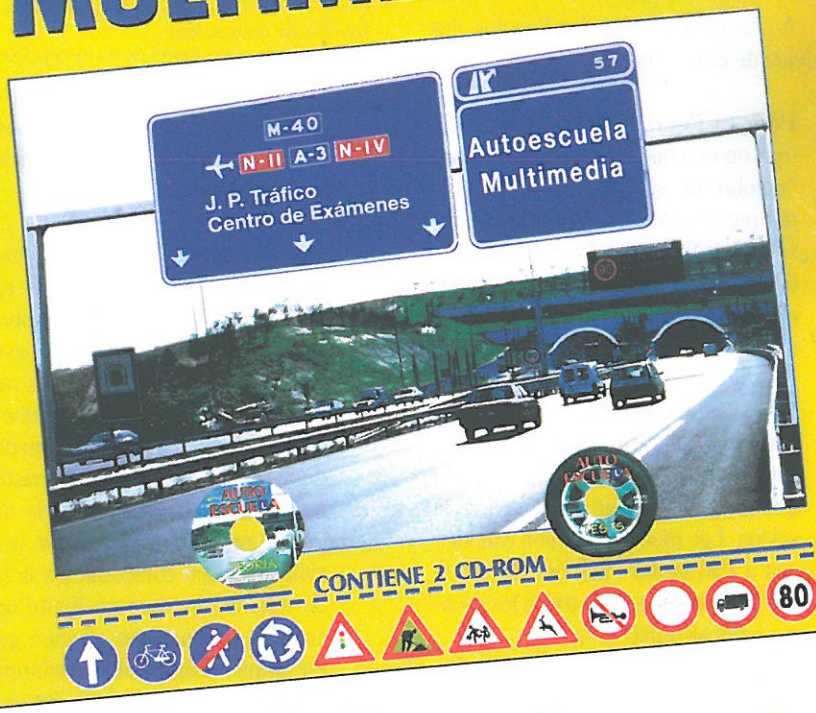
DirectPlay creará aplicaciones Peer-to-Peer

El primer paso para utilizar *DirectPlay* consiste en seleccionar qué proveedor de servicios se va a utilizar. El proveedor de servicios determina qué tipo de red o protocolo se utilizará para las comunicaciones. El protocolo puede variar de tipo dependiendo del entorno que use para enviar y recibir información:

- TCP/IP para Internet.
- IPX para red de área local.
- Conexión por cable en serie para comunicarse entre dos ordenadores.

El proveedor de servicios se utiliza para crear una conexión a un punto de la red. El usuario puede necesitar introducir información adicional para crear dicha conexión o ser la misma aplicación la que especifique los parámetros de conexión.

AUTO ESCUELA MULTIMEDIA 98



2 CD-Rom por 2.995 ptas.

Sácate el carnet del ordenador.

Nueva versión completamente actualizada. (Incluye tests y autoexámenes)

ABETO editorial

c/ Aragoneses, 7 • 28108 Alcobendas (Madrid)
Tel.: (91) 661 42 11* • Fax: (91) 661 43 86
http://www.abetoed.es

COMUNICACIONES

Una aplicación puede gestionar sus propios datos utilizando el modelo *Cliente/Servidor* o *Peer-to-Peer*, pero cambiará el modo en que se gestiona el estado de la sesión *DirectPlay* activa.

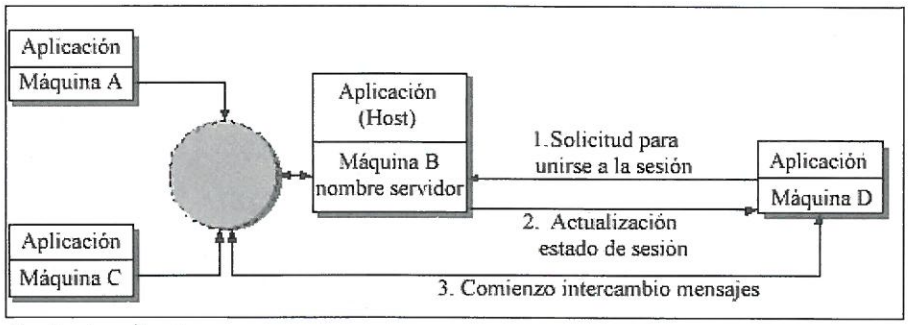
A continuación, se detallan los dos modelos de comunicación:

- **Peer-to-Peer**: *en* una sesión de este tipo, un ordenador es designado como "nombre de servidor". Este ordenador regula a los otros intentando unirse a la sesión, actualizando el estado de la sesión a las nuevas máquinas que se han unido, y genera tantos números de identificación (ID) como jugadores y grupos se hayan creado. Más allá de todo esto, el "nombre de servidor" es otro miembro en la sesión y ejecuta la misma aplicación que todos los demás miembros de la sesión. Los mensajes no son conducidos a través del "nombre de servidor", y éste no genera todos los mensajes del sistema.

  DirectPlay, automáticamente utiliza las funciones del "nombre de servidor", las cuales no son expuestas a la aplicación en ningún momento. La aplicación *DirectPlay* que se ejecuta como "nombre de servidor" se denomina *Host* de la sesión. Únicamente la aplicación que ejerce como *Host* puede cambiar los datos que describen la sesión. Cada máquina conectada a la sesión tiene acceso completo a la lista de jugadores y grupos y puede enviar y recibir mensajes de cualquier otro jugador incluido en la sesión.

  La dirección de red de la sesión se corresponde con la dirección de red del "nombre de servidor". Cuando un ordenador necesita unirse a la sesión, éste envía la solicitud al "nombre de servidor". En respuesta, este último actualiza el estado de la sesión a los nuevos ordenadores incorporados. Si el "nombre de servidor" deja la sesión, un nuevo ordenador es elegido para sustituirle en sus funciones.

  Las sesiones *Peer-to-Peer*, generalmente tienen una limitación en el número de ordenadores que pueden participar. Cada cambio en el estado de los ordenadores (como los movimientos de los jugadores) debe ser transmitido a todos los ordenadores de la sesión. Debido a que existe un límite sobre qué cantidad de datos puede recibir un ordenador (especialmente con conexiones a la línea de teléfono), existe un límite sobre cuántos ordenadores pueden generar datos. Minimizando la cantidad y la frecuencia de los datos intercambiados, se puede incrementar el número de ordenadores que pueden coexistir en una sesión antes de que el rendimiento decaiga.



Cuadro de aplicaciones.

Un servidor multicast es un ordenador que reenviará un mensaje a múltiples destinos

En una comunicación *Peer-to-Peer*, los ordenadores se comunican directamente unos con otros. Esto significa que los mensajes no son enviados a través de un ordenador que ejerza de intermediario. Cuando *DirectPlay* necesite enviar el mismo mensaje a un grupo de ordenadores, intenta utilizar cualquier recurso *multicast* del proveedor de servicios. Si el *multicast* no está soportado, los mensajes serán enviados mediante el recurso *unicast* (los mensajes son enviados a cada máquina de forma individual).

Esta técnica puede ser insuficiente en sesiones con un largo número de jugadores (más de cuatro). Incluyendo un servidor *multicast* a la sesión, este problema se puede aliviar de forma considerable. Un servidor *multicast* es un ordenador que reenviará un mensaje a múltiples destinos. Un ordenador que necesite transmitir un mensaje a los demás ordenadores en la sesión, puede enviar un mensaje al servidor *multicast* para que se encargue de enviar los mensajes individuales a los demás equipos.

Esta técnica es más eficiente debido a que el servidor *multicast* está conectado a la red a través de un enlace de alta velocidad y puede enviar mensajes *unicast* más rápidamente que cualquier equipo individual. El concepto de este incremento de velocidad en servidores *multicast* radica en el uso de líneas RDSI en lugar de líneas telefónicas básicas.

En el servidor *multicast*, los mensajes de jugador a jugador son enviados directamente a la máquina destino. Además, los mensajes de grupo son enviados al servidor *multicast*, que a su vez, los reenvía a las demás máquinas destino.

DirectPlay soporta la creación de un servidor *multicast* en una sesión. Si el "nombre de servidor" está en un enlace de alta velocidad, puede elegir ser un servidor *multicast* en la sesión. La inclusión de un servidor de este tipo en una sesión sólo cambia la forma en que son enviados los mensajes de grupo.

- **Cliente/Servidor:** En una sesión de este tipo, un ordenador es designado como servidor. Como el "nombre de servidor" utilizado en *Peer-to-Peer*, este ordenador responde a las solicitudes, regula los equipos intentando unir la sesión, actualizando el estado de la misma a los nuevos ordenadores que se han unido, y genera tantos números de identificación (ID) como jugadores y grupos se hayan creado. En diferencia con el "nombre de servidor" de Peer-to-Peer, todos los mensajes en la sesión son enviados a través del servidor.

En las comunicaciones Cliente/Servidor, cada máquina comunica con el servidor

El ordenador que hace de servidor ejecuta una versión especial de la aplicación (aplicación servidora) que puede ejercer un control total sobre el juego, actualizar su estado acorde con las acciones que efectúan los clientes, y notificar los eventos relevantes de los clientes individuales. La aplicación servidora crea un jugador especial llamando "jugador servidor".

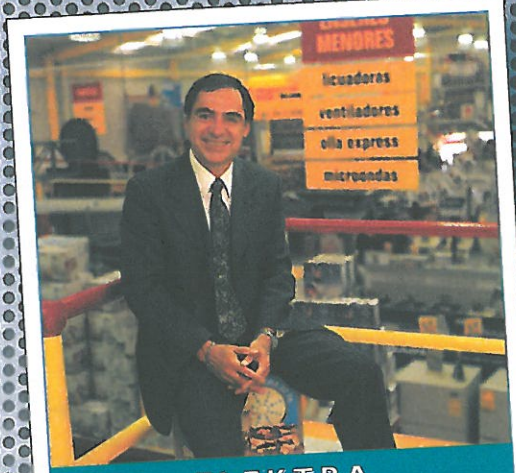
Cualquier ordenador cliente puede enviar o recibir un mensaje directamente del "jugador servidor". Cuando el "jugador servidor" recibe un mensaje, éste puede enviar mensajes a otros "jugadores clientes" para informarles de un cambio de estado.

En las comunicaciones *Cliente/Servidor*, cada máquina comunica única y exclusivamente con el servidor y éste puede reenviar mensajes a los otros clientes. Los ordenadores clientes ejecutan versiones de la aplicación que puedan mantener el estado local, actualizándolo en respuesta a los mensajes enviados y recibidos del servidor cuando una acción local ha ocurrido.

Podemos implementar una aplicación *Cliente/Servidor* de dos modos:

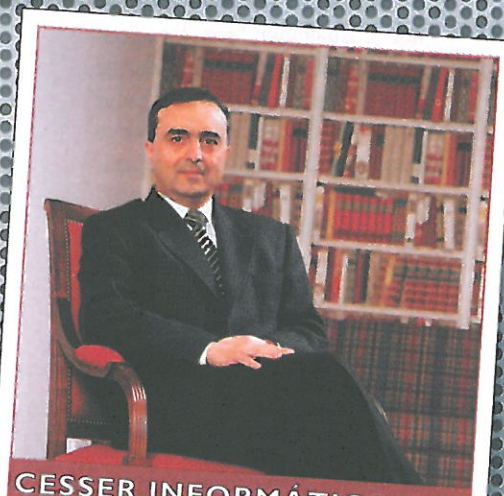
1. El primero de ellos consiste en un híbrido de sesiones *Peer-to-Peer* y *Cliente/Servidor*, donde cada aplicación cliente tiene la lista completa de

IBM
Soluciones para nuestro pequeño mundo



ELEKTRA
Gustavo Vega Vázquez

La mensajería de **MQSeries para Windows NT** facilita a Elektra el seguimiento de cada venta en los 500 establecimientos, desde su oficina central en Ciudad de Méjico, utilizando la conexión por satélite tan eficientemente como el espacio en sus estanterías.



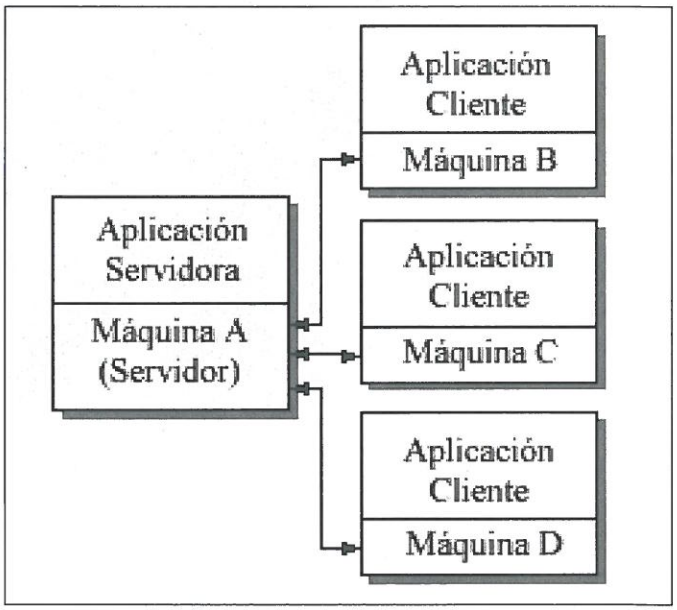
CESSER INFORMÁTICA, S.L.
Renato Aquilino

"**DB2 Universal Database sobre plataforma Windows NT** es el motor elegido para impulsar nuestra nueva línea de productos Visual Gesmarket. Porque gracias a su capacidad multimedia, que ofrece posibilidades de integración de todo tipo de datos, su escalabilidad, que nos permite implantar nuestras soluciones en empresas de todos los tamaños, y su amplia conectividad, DB2 ofrece el alto nivel de prestaciones que nuestros clientes buscan en su software presente y futuro."

Miles de empresas ya saben cómo IBM optimiza Windows NT.

La familia de productos de software de IBM para Windows NT contiene todo lo que usted necesita para diseñar y desarrollar las nuevas aplicaciones, imprescindibles para que su negocio sea más competitivo. Todas las funciones integran sin problemas con sus sistemas centrales, sacando mayor provecho de sus ordenadores bajo Windows NT.

Lotus Domino es marca de Lotus Development Corp. Windows NT es marca de Microsoft Corp. DB2, MQ Series y Visualage son marcas de IBM Corporation. © 1998 IBM Corp.



Esquema de aplicación Cliente/Servidor.

jugadores y grupos disponibles (incluyendo el "jugador servidor"). Cualquier "jugador cliente" puede enviar un mensaje directamente a cualquier jugador de la sesión, y cada jugador o grupo puede tener datos asociados con el "jugador cliente" (como nombre o datos remotos) que serán actualizados sobre todos los ordenadores cuando cambien. Todos los beneficios del estado de propagación de *Peer-to-Peer* están activos para utilizarlos, además de un "jugador servidor". De todas formas, no podemos controlar la propagación de datos de jugador a jugador. Dependiendo de la frecuencia con la cual los jugadores y grupos son creados y destruidos o cambian sus datos, los ordenadores clientes podrían ser sobrecargados con cantidad de mensajes "en cola". Este tipo de sesión no es escalable más allá del límite de 16 jugadores. Esto no es muy diferente de una sesión *Peer-to-Peer*, excepto en que tiene un "jugador servidor", todos los mensajes son dirigidos a través del servidor y que el "jugador servidor" recibirá una copia de todos los mensajes que pasen a través del servidor.

2. El segundo modo es puramente *Cliente/Servidor*. Aquí, cada aplicación cliente ve sólo el "jugador servidor" y sus jugadores locales en la sesión. Un "jugador cliente" puede enviar mensajes y recibirlos sólo del "jugador servidor". La aplicación servidora puede ver todos los jugadores sobre todos los clientes. Cuando el servidor recibe un mensaje de un cliente, éste puede decidir cómo actualizar el estado de la aplicación principal y qué clientes deben informar de su actualización. En este modo, la aplicación cliente debe manejar la lista de jugadores y los datos asociados con cada jugador.

Los lobby's pueden ser activados para permitir más servicios al usuario

Al igual que en las sesiones *Peer-to-Peer*, la dirección de red de la sesión se corresponde con la dirección de red del servidor. Cuando un ordenador necesite unirse a una sesión, éste envía la solicitud al servidor. En respuesta, el servidor actualiza el estado de la sesión a los nuevos ordenadores incorporados. Si el servidor deja la sesión de forma repentina, ésta también concluirá desuniendo a los jugadores.

LOBBY´S

Un servidor *Lobby* en *DirectPlay* es un lugar de común encuentro en una red que sigue la pista de las sesiones en progreso y los usuarios que estén conectados al servidor. Los usuarios pueden navegar alrededor del servidor *lobby* para encontrar áreas de interés. En cualquier lugar del mismo, el usuario puede utilizar el servicio IRC para comunicarse con otros usuarios, unirse a las sesiones en progreso o reunir un grupo de jugadores para comenzar una nueva sesión.

La mayor ventaja de un servidor *lobby* reside en que actúa como una central donde los usuarios pueden encontrar sesiones y otras personas para interactuar entre ellos. El servidor *lobby* maneja todas las direcciones de los jugadores y sesiones, y automáticamente puede lanzar aplicaciones y conectar a los usuarios con la dirección correcta sin intervención por parte de ellos. El manejo de sesiones, jugadores, y sus direcciones de Red es especialmente útil en Internet, donde generalmente los usuarios no pueden encontrar oponentes para jugar.

Los *lobby's* pueden ser activados para permitir más servicios al usuario como: torneos, perfiles personales, noticias, actualizaciones de software, etc. Una aplicación que detecte *lobby's* es aquella que ha sido desarrollada para interactuar con estos servidores. Hay dos tipos de *lobby's* que un usuario puede experimentar:

1. El *lobby* externo es una aplicación cliente en el que su único fin es interactuar con un servidor *lobby* y los usuarios conectados a él. Cuando se comienza o se une una aplicación a la sesión, el cliente *lobby* lanza la

aplicación en un proceso separado y envía toda la información necesaria para establecer una conexión con la sesión.

2. El *lobby* interno es una interfaz de usuario que está integrado en la aplicación por sí mismo. Este método es más difícil de implementar debido a que la aplicación debe crear dicho interfaz de usuario tanto para el *lobby* como para el juego.

El creador de aplicación servidora *lobby* debe escribir el proveedor *lobby* que reside en el ordenador cliente. La aplicación llama al API *DirectPlay* y los servicios de la DLL del proveedor *lobby* para comunicarse con el software del servidor *lobby*. Como mínimo, este último debe ser capaz de localizar a todos los usuarios conectados a él, organizar a los jugadores formando grupos y sincronizar el lanzamiento de una aplicación en la sesión.

"Sesión lobby" es una conexión con un servidor donde los clientes y el servidor no han sido específicamente escritos para operar entre ellos

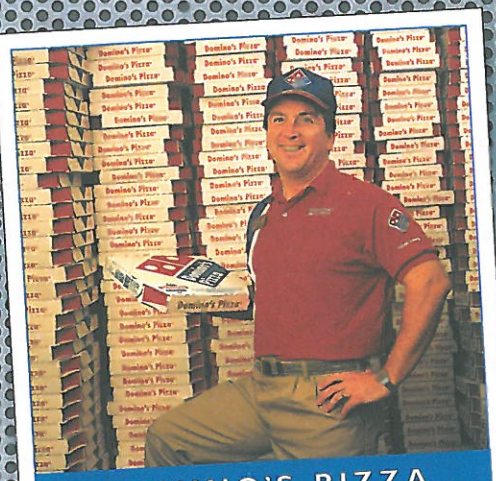
DirectPlay define un nivel común de funcionalidad para todos los servidores *lobby*. Cualquier aplicación cliente puede conectarse a cualquier servidor *lobby* y operar correctamente.

No existen diferentes API's para comunicarse con el servidor *lobby* de *DirectPlay*. La interacción con este servidor ha sido abstraída para que el mismo interfaz *DirectPlay3* usado para comunicarse con una sesión de la aplicación, pueda ser utilizado para contactar con el servidor *lobby*. Los mensajes y métodos adicionales han sido agregados a la interface para soportar la funcionalidad adicional que requiere un servidor *lobby*.

SESIONES LOBBY

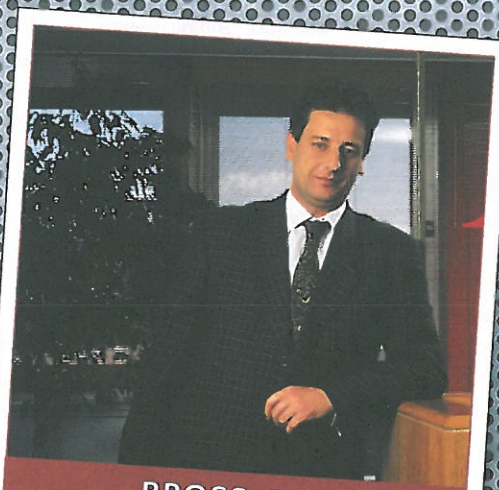
Una sesión *lobby* es muy parecida a una sesión *Cliente/Servidor* de *DirectPlay*. El término "sesión *lobby*" se refiere a una conexión con un servidor de estas características donde los clientes y el servidor no han sido específicamente escritos para operar entre ellos. El término "sesión de la aplicación" se refiere a la tradicional sesión *DirectPlay* en la cual todos los clientes y el servidor han sido específicamente escritos para operar entre ellos.

IBM
Soluciones para nuestro pequeño mundo



DOMINO'S PIZZA
David Melton

Franquiciados como David Melton no necesitan aprender HTML para trabajar con **Lotus Domino bajo plataforma Windows NT**. Ahora con su intranet es más rápido y fácil compartir ideas y trabajar en equipo contra la competencia.



PROSS, S.A.
Alfredo Giner

"El principal cliente de PROSS es la Banca y gracias a **DB2 Universal Database para Windows NT** se consigue una distribución de las aplicaciones en las bases de datos de diversos servidores, con una notable mejora del rendimiento y la fiabilidad que necesitan nuestros clientes. DB2 ha introducido el tratamiento de imágenes y sonido en transacciones tradicionales haciendo de su uso y mantenimiento un proceso sencillo e intuitivo."

Y están preparadas
para e-business.

Todo el software de IBM para Windows NT está preparado para Internet. Así, todos los usuarios, proveedores y clientes, desde cualquier lugar, podrán acceder a las funciones más importantes, independientemente de la plataforma en la que trabajen. Si desea información detallada de los productos de software de IBM, visítenos en www.software.ibm.com/nt

e-business

Lotus Domino es marca de Lotus Development Corp. Windows NT es marca de Microsoft Corp.
DB2, MQ Series y Visualage son marcas de IBM Corporation. © 1998 IBM Corp.

Una sesión *lobby* de *DirectPlay* se utiliza para representar una conexión a un servidor *lobby*. Como en una aplicación *DirectPlay*, el primer paso para comunicarse con este tipo de servidor consiste en seleccionar qué proveedor *lobby* se va a utilizar y qué servidor *lobby* se empleará para conectarse. En *DirectPlay*, existen varios métodos que efectúan estas tareas y están implementados en el interface *DirectPlay3*:

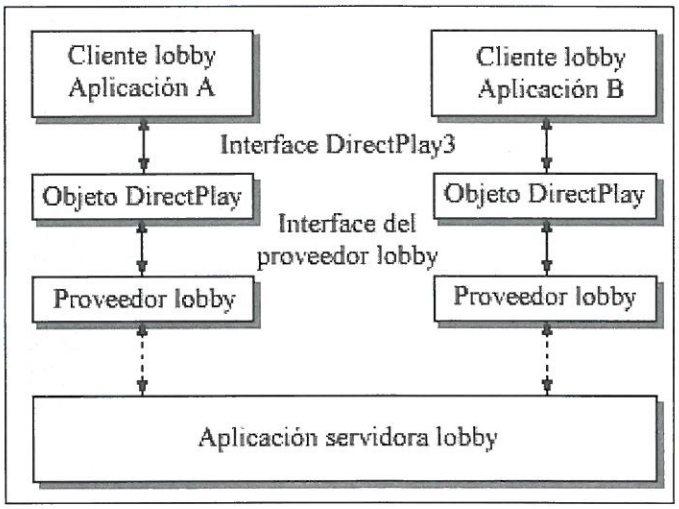
- EnumConnections.
- InitializeConnection.

El cliente *lobby* utiliza los mismos métodos para manejar las sesiones que una aplicación cliente para localizar y unirse a una sesión *lobby*; por ejemplo, *EnumSessions*, *Open*, *SecureOpen*, y *Close*. Al unirse a una sesión, obtenemos acceso en la aplicación cliente a toda la información del servidor *lobby*, permitiendo al usuario interactuar con otros jugadores de dicho servidor.

Como una sesión de aplicación *DirectPlay*, el jugador es la entidad básica en la sesión *lobby*. Cada jugador representa un usuario sobre un ordenador cliente conectado al servidor *lobby*. También existe un jugador servidor representando el lobby. Una vez que el lobby se conecta al servidor, (uniéndose a la sesión) debe establecer la presencia del usuario creando un jugador y sumándolo al grupo inicial antes de que pueda comenzar la comunicación con el servidor y los demás jugadores. De hecho, los otros usuarios conectados a la sesión no detectarán la presencia de nuevos usuarios hasta que este hecho ocurra.

Los métodos usados en sesiones de aplicación son utilizados de igual forma para los jugadores; por ejemplo, *EnumPlayers*, *CreatePlayer*, *DestroyPlayer*, *GetPlayerName*, *SetPlayerName*, y *GetPlayerCaps*.

Nota: *Al contrario que una sesión de aplicación, los métodos Send y Receive, generalmente no suelen ser utilizados para intercambiar mensajes con otros jugadores.*



Esquema lobby.

Las funciones comunes que requieren comunicaciones entre clientes y servidor tienen nuevos métodos creados exclusivamente para ellos. *DirectPlay* determina el formato de los mensajes. El destinatario recibe un mensaje de sistema almacenando su contenido en una estructura de datos bien definida. Por ejemplo, el método *SendChatMessage* envía un mensaje de texto de tipo *chat* a otro jugador.

Los grupos toman más relevancia en sesiones *lobby* porque son utilizados para definir la organización de todo el servidor y lanzar sesiones de las aplicaciones. Podemos crear una jerarquía completa de los grupos para manejar y organizar todos los jugadores y sesiones que el servidor localice.

Los grupos son clasificados de dos formas diferentes. Un grupo estándar puede contener jugadores y otros grupos, el cual es llamado *Room* (habitación). Un *Room* es utilizado como lugar de encuentro para que los jugadores interactúen entre sí. El *Room* contiene enlaces a otros *Rooms* y al segundo tipo de grupo: *Staging area*.

El grupo *Staging area* típicamente contiene sólo jugadores, y se utiliza para formar jugadores y lanzar nuevas sesiones. Una vez que la sesión ha sido lanzada, el *Staging area* puede instalarse para que los nuevos jugadores puedan unirse a la sesión en progreso.

Los grupos del *Room* están organizados según una estructura jerárquica. Unos cuantos grupos de alto nivel son creados, y a su vez, otros grupos son creados dentro de los anteriores.

El cliente *lobby* puede enumerar tanto los grupos de alto nivel como los contenidos en ellos. El servidor *lobby* puede organizar los grupos en una estructura Web. Una vez que un grupo es creado, podemos agregar éste a otro grupo, el cual crea un enlace entre los dos. Cuando un cliente *lobby* enumera los grupos contenidos en uno, se devuelven los subgrupos y grupos enlazados.

Mediante el uso de Proveedores, utilizaremos los soportes físicos de red que tengamos instalados

Los jugadores pueden navegar a través del espacio del servidor borrándose ellos mismos del grupo en el que se encuentran y agregándose a uno nuevo. En cualquier momento, el método *EnumGroupsInGroup* puede ser llamado para determinar qué grupos están enlazados al grupo actual.

PROVEEDORES

La interfaz DirectPlay se utiliza para enviar y recibir mensajes mediante servicios de alto nivel para el manejo de jugadores, manejo de datos y propagación, servicios *lobby*, etc. El proveedor de servicios abastece los servicios de comunicación de la red requeridos por *DirectPlay*. Los servicios *Online* y operadores de red pueden suplir los proveedores de servicios utilizando hardware especial, protocolos, y recursos de red. Un proveedor de servicios puede ser simplemente una capa entre *DirectPlay* y cualquier transporte existente como el *Winsock*, o puede utilizar recursos especializados sobre un servicio *Online* como los servidores *multicast*. Microsoft incluye cuatro proveedores de servicios genéricos en DirectPlay:

1. **Head-to-head modem (TAPI)**: comunicaciones de módem a módem utilizando TAPI (API de telefonía).

   La creación de una sesión, (utilizando el método *Open method*) hace que aparezca una caja de diálogo, preguntando al usuario sobre qué módem se espera la llamada. El método *EnumSessions* visualizará una caja de diálogo preguntando al usuario a qué número de teléfono llamar y qué módem utilizar. Una vez se introduce la información, *DirectPlay* marcará en el módem e intentará encontrar sesiones albergadas por el ordenador del otro usuario.

   La lista actual de modems disponibles puede ser obtenida del proveedor de servicios inicializando y llamando al método con una identificación (ID) del jugador igual a **0**. La dirección devuelta por *DirectPlay* contendrá un *chunk* de datos con los nombres de los modems en formato ANSI y Unicode. La lista de modems es una serie de cadenas terminadas en caracteres nulos con una cadena de longitud 0 al final de la misma.

2. **Conexión serie**: un enlace serie es utilizado para comunicarse con otros ordenadores a través de los puertos serie. Cuando se crea una sesión (utilizando el método *Open*), aparece un cuadro de diálogo preguntando al usuario la configuración del puerto serie. Debemos configurar el puerto serie de la misma forma en los dos ordenadores que se comuniquen.

3. **Internet TCP/IP**: este proveedor utiliza *Winsock* para comunicarse en Internet o redes de área local utilizando el protocolo *TCP/IP*. Éste usa paquetes *UDP* (Protocolo de Datagrama de Usuario) para los mensajes no garantizados y *TCP* para los garantizados. Un único ordenador puede albergar múltiples sesiones *DirectPlay* utilizando *TCP/IP*.

Los servicios Online pueden suplir los proveedores de servicios

   Cuando se utiliza el método *EnumSessions*, *TCP/IP* visualizará una caja de diálogo preguntando al usuario la dirección IP de la sesión. El usuario debe introducir la dirección IP del ordenador que albergará la sesión que va a ser unida. Si el ordenador tiene un nombre (como laserscreen.com), puede ser utilizado en lugar de su dirección IP, y *DirectPlay* buscará el nombre de dominio de sistema (*DNS*: *Domain Name System*) para encontrarlo. *EnumSessions* devolverá las sesiones que alberga el ordenador. El usuario puede dejar la dirección en blanco y pulsar **OK**. En este caso, *DirectPlay* transmitirá un mensaje de búsqueda de sesiones activas en ese momento.

   Nota: Un usuario de Windows 95 puede determinar su dirección IP eligiendo **Ejecutar** desde el botón **Inicio** y escribiendo **WINIPCFG**. En Windows NT se ejecuta **IPConfig** desde la línea de comandos. Si el usuario está conectado tanto a una LAN como un proveedor de servicios de Internet (ISP), el ordenador puede tener dos direcciones IP y se debe seleccionar la correcta. Muchos ISP's asignan direcciones IP dinámicas que cambian cada vez que el usuario se conecta.

   Si al método le damos el nombre inicial de *InitializeConnection*, o *SetConnectionSettings* seguido por la llamada a conectar, suprimimos la dirección IP del proveedor de servicios por la de *DirectPlay*.

   El proveedor *TCP/IP* generalmente no trabaja a través de *firewalls*. Incluyendo el soporte *lobby* de *DirectPlay*, podemos eliminar la necesidad de los usuarios de introducir la dirección IP si comienzan el juego desde un servidor *lobby*.

   Nota: *Un usuario de Windows 95 puede configurar su ordenador para visualizar o no el cuadro de diálogo que requiere información de conexión cuando DirectPlay intenta completar conexiones vía TCP/IP. Para suprimir estos diálogos, realizaremos los siguientes pasos:*

   - Abrir el *Panel de Control*.
   - Hacer doble clic en el icono *Internet*.
   - Elegir la pestaña Conexión.
   - Desactivar el *Checkbox* "Conectarse a Internet cuando sea necesario" .

4. **IPX** : el proveedor de servicios *IPX* el *Winsock* para comunicarse sobre redes de área local (LAN) usando el protocolo de Intercambio de Paquetes de Internet (*IPX*). El proveedor de servicios sólo soporta mensajes no garantizados. Un ordenador sólo puede albergar una sesión de *DirectPlay* utilizando *IPX*.

   IPX siempre usa una transmisión para encontrar sesiones en la red; el

Rellena este cupón

CORRE. CADA 3 SEGUNDOS MUERE UN NIÑO EN EL TERCER MUNDO.

☐ SI, DESEO RECIBIR MAS INFORMACION SIN COMPROMISO.

Nombre .....

Dirección .....

Localidad ..... Provincia .....

C.P. ..... Tel. .....

C/ Tutor, 27. 28008 Madrid. C/ Balmes, 32, 3º. 08007 Barcelona.

718 **Tel.: 902 402 404**

Ayuda en Acción

Practica el mejor ejercicio para tu corazón: la solidaridad. En Ayuda en Acción, necesitamos que la ejercites para poder impulsar nuestros Proyectos de Desarrollo Integral en el Tercer Mundo. Corre. Rellena este cupón y contribuirás a llevar la esperanza a quien más lo necesita.

Apadrina un niño. Tel.: 902 402 404

Desde 1981
trabajando con el tercer mundo

Ayuda en Acción

método *EnumSessions* no visualizará ninguna caja de diálogo requiriendo direcciones IP. Una vez que la sesión se ha establecido, los paquetes son enviados (no transmitidos) directamente entre los ordenadores.

La interfaz *DirectPlay* oculta las complejas tareas requeridas para establecer comunicaciones arbitrarias enlazando la implementación de las funciones del proveedor de servicios dentro de él. Una aplicación que utilice *DirectPlay*, sólo necesita saber los detalles concernientes al rendimiento y características de la red virtual presentada por *DirectPlay*. No es necesario saber si es el módem, la tarjeta de red, o el servicio *online* el que provee el medio de comunicación.

DirectPlay se une dinámicamente a cualquier proveedor de servicios instalado en el sistema del usuario. La aplicación interactúa con el objeto *DirectPlay*. Éste, a su vez, interactúa con uno de los proveedores de servicios disponibles, y el proveedor seleccionado lo hace con el transporte, protocolo y otros recursos de la red.

El API *DirectPlay* se presenta a la aplicación a través de ciertos interfaces *COM*. Las sesiones de la aplicación pueden hablar a alguien o algo de la red a través de los interfaces *DirectPlay3* y *DirectPlay3A*.

El primero utiliza cadenas Unicode en todas las estructuras *DirectPlay*, mientras que el segundo usa cadenas ANSI. *DirectPlay, DirectPlay2, y DirectPlay2A* aún existen para la compatibilidad de versiones anteriores de *DirectX*. Los clientes *lobby* pueden hablar a otra aplicación en la misma máquina a través de los interfaces *DirectPlayLobby2 y DirectPlayLobby2A*.

NUESTRO PROYECTO

En el siguiente artículo comenzaremos la programación de *DirectPlay* mediante un sencillo programa tipo 3 en raya utilizando cualquier proveedor de servicios que tengamos instalado en nuestro equipo.

Lógicamente, será necesaria la colaboración de otro ordenador para establecer la conexión física a la red entre ambos equipos.

Para que el programa funcione correctamente, será necesaria la instalación del acceso telefónico a redes con el protocolo TCP/IP o IPX para conexión entre tarjetas de red. La conexión de equipos mediante enlace directo de módem a módem no es muy aconsejable debido al incremento en las tarifas telefónicas.

Acceso Remoto (RAS)

Enrique Díaz Trobo (enriqued@jet.es)

Abordamos en este número el Acceso y, por tanto, el trabajo remoto. Seguramente muchos de nuestros usuarios comienzan a demandar una conexión que les permita conectarse a la oficina sin tener que salir de casa. No todos los trabajos son presenciales de tal forma que en este artículo veremos cómo aprovechar las conexiones remotas.

Aunque resulte increíble, el teletrabajo ya es una realidad para un número cada vez mayor de usuarios. Para el trabajador representa muchas ventajas y algunos inconvenientes. Respecto a las empresas, el saldo resulta más favorable aún, puesto que prácticamente todo son ventajas. Por una parte, el equipo informático así como su mantenimiento, suelen ser responsabilidad del trabajador, de tal forma que la empresa se ahorra inversiones en equipos y lo que es aún más relevante, evita las ingratas labores de mantenimiento. También se ahorra el gasto de consumo eléctrico y el lugar que ocupa el teletrabajador en la oficina. De hecho, es posible instalar una empresa donde trabajan cientos de empleados en un reducido espacio. Tan sólo es necesario el espacio que ocupen los servidores.

Para el teletrabajador también representa alguna ventaja, como por ejemplo ahorrarse los gastos de desplazamiento hasta la empresa o ajustar el propio horario a sus necesidades. Por el contrario, este tipo de facilidades es muy propensa a la realización de trabajos sin contrato, al no existir una vinculación física entre empresa y trabajador.

En este artículo explicaremos la forma de preparar Windows NT para aceptar llamadas y conectar usuarios remotos. Si deseamos tener un buen número de usuarios remotos, deberemos también instalar algún hardware adicional, como por ejemplo algún *router* y, por supuesto, contratar líneas RDSI.

Windows NT implementa las conexiones remotas mediante *RAS* (*Remote Access Server*: Servidor de Acceso Remoto). Por supuesto, ésta no es la única ni la mejor posibilidad, pero tiene la ventaja de ser unos de los componentes de serie de Windows NT y, por tanto, es gratis.

MÉTODOS DE COMUNICACIÓN DE RAS

El *RAS* soporta clientes que utilicen el protocolo *PPP* que no sean Microsoft pero utilicen *TCP/IP, IPX o NetBEUI*; y emplea los siguientes métodos de conexión:

- *Módem analógico*. *RAS* soporta una gran variedad de modems que pueden ser configurados para soportar llamadas entrantes y salientes a través de la red telefónica conmutada. Utiliza tres tipos de protocolos de módem:
  - *Protocolo SLIP*. El Protocolo de Internet de Línea de Serie es un protocolo antiguo desarrollado para el entorno UNIX. Trabaja

sin control de errores, control de flujo ni seguridad, pero consigue a cambio un buen rendimiento con pequeños bloques de datos. *RAS* soporta *SLIP* para llamadas salientes permitiendo acceder a equipos NT e Internet.

- *Protocolo PPP*. El Protocolo Punto a Punto es un protocolo *SLIP* mejorado con control y recuperación de errores. *RAS* soporta *PPP* para llamadas salientes y entrantes.
- *Protocolo RAS*. Es un protocolo desarrollado por la propia Microsoft para soportar *NetBIOS* en todas las versiones de RAS.

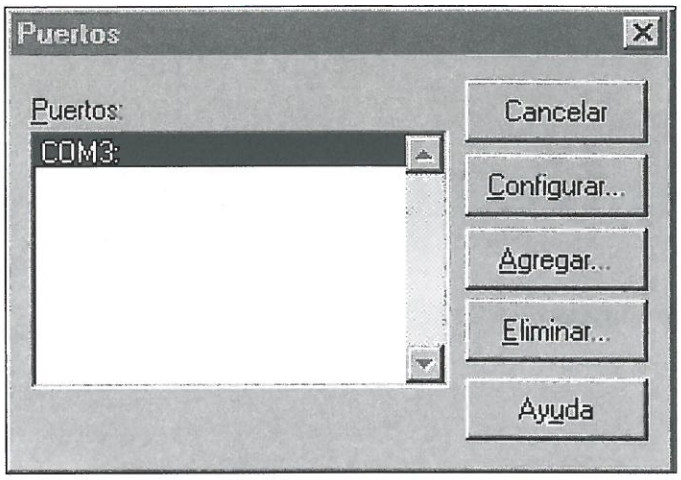


Figura 1.

- *Módem Nulo*. Es un cable que conecta dos ordenadores a través de sus puertos de serie RS232C. De esta manera se puede conectar un ordenador a una red, utilizando *RAS* y sin necesidad de tarjetas de red.

- *RDSI*. La Red Digital de Servicios Integrados ha supuesto un gran avance en comunicaciones digitales. Utiliza dos *routers*, uno en cada extremo, y una línea telefónica digital para enviar datos a 64 Kilobits por segundo, cifra que puede doblarse hasta 128 Kb por segundo utilizando el segundo canal.

- *X25*. Este es un método antiguo y lento, pero eficaz. Permite conectarse a una red utilizando dos *routers*, uno en cada extremo, y un sistema basado en paquetes que viaja a través de la red telefónica. Su utilización es recomendable para distancias cortas y poco volumen de datos.

SEGURIDAD

RAS soporta un nivel muy alto de seguridad, ya que las llamadas entrantes de usuarios tienen que ser autentificadas por Windows NT Server antes de permitir su entrada en la red. Además, puede ser configurado para trabajar con devolución de llamada, es decir, cuando un usuario efectúa una llamada al servidor RAS, éste cuelga la llamada para, a continuación, efectuarla a un teléfono previamente establecido. De esta manera, aunque alguien llegara a conocer la contraseña de un usuario, no podría realizar un acceso remoto, ya que el servidor RAS llamaría al teléfono previamente establecido para el verdadero usuario.

Así, también se produce un ahorro de dinero para los usuarios, ya que las llamadas corren por cuenta de la empresa. Otro sistema de seguridad que incorpora es la codificación de datos, que se puede utilizar para proteger los datos que se envían por la línea telefónica.

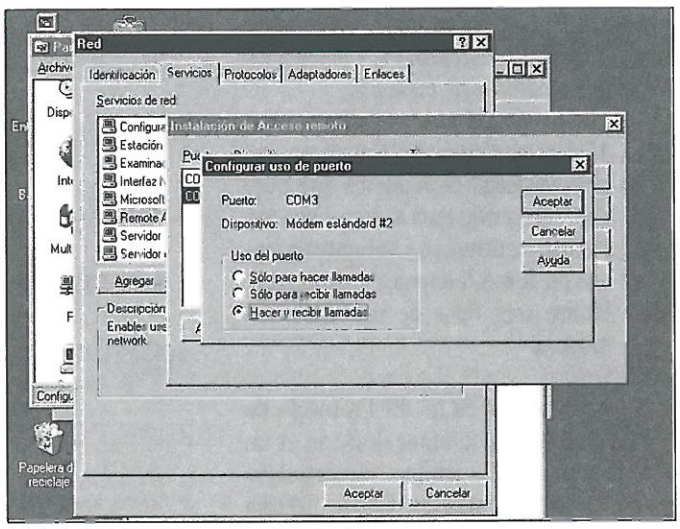


Figura 2.

INSTALAR Y CONFIGURAR EL RAS

Para instalar y configurar RAS iniciaremos la sesión como un usuario miembro del grupo *Administradores* y seguiremos los siguientes pasos, que se exponen a continuación:

1. Configurar los puertos de serie.
2. Instalar uno o varios modems.
3. Instalar el software de *RAS*, si no se instaló durante la instalación de Windows NT.
4. Configurar los puertos *RAS*.
5. Configurar las cuentas de usuario que tendrán acceso remoto.

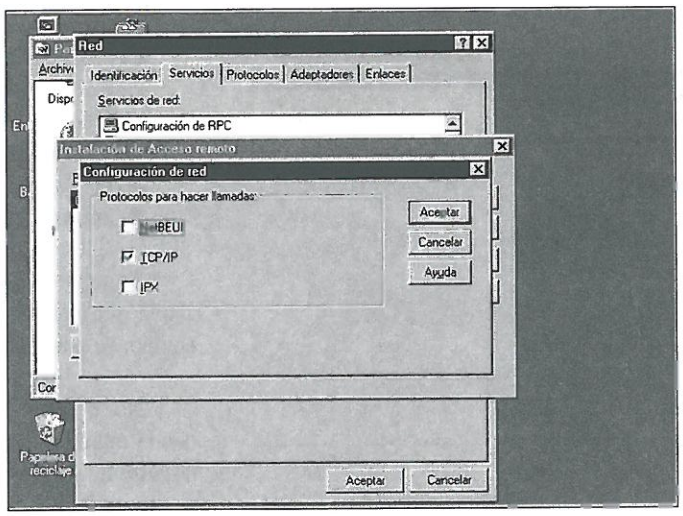


Figura 3.

CONFIGURACIÓN DE LOS PUERTOS DE SERIE

Para configurar los puertos de serie pincharemos sobre el icono **Puertos** del *Panel de control.* Veremos una pantalla similar a la de la figura 1. Seleccionaremos aquellos puertos que nos interese añadir al *RAS*, cambiando aquellos valores de la configuración que nos interese. También podremos agregar y eliminar puertos desde esta pantalla.

El RAS da soporte a 256 puertos.

Un problema muy común que suele presentarse en algunos equipos es la imposibilidad de utilizar los puertos COM1 y COM3 y COM2 y COM4 simultáneamente. Como Windows NT da soporte a 256 puertos, podemos evitar este problema configurando, por ejemplo, COM3 como COM 253. Los puertos que reconoce la BIOS (COM1 y COM2) es mejor dejarlos como están ya que cambiar sus valores seguramente dará problemas. Por ejemplo, tenemos pocas probabilidades de que funcione configurar COM1 como COM251.

CONFIGURAR EL MÓDEM

Una vez instalado físicamente el módem, tendremos que configurarlo para que funcione con el *RAS*. Para ello ejecutaremos pincharemos en el icono Modems del *Panel de control,* lo que hará aparecer el asistente de instalación asociado.

Si tenemos disquete del fabricante lo mejor será elegir la opción de *No detectar el Módem.* Posteriormente se nos pedirá elegir el módem adecuado de una lista de fabricantes y modelos, donde una de las opciones posibles es la de obtener el controlador de un disco del fabricante.

Otro de los modems posibles es el cable de serie de acceso telefónico a redes entre dos PC's. Esto es, el *Null Modem* o módem nulo (lo podréis encontrar entre los modelos de módem estándar).

Entre las propiedades del módem aparece una denominada *Conectar únicamente a esta velocidad.* Esta opción evita que el módem se conecte a velocidades inferiores a la especificada.

Al añadir un módem al *RAS* podemos especificar el uso que deseamos para el puerto, esto es, *Sólo para hacer llamadas, Sólo para recibir llamadas* o *Hacer y recibir llamadas* (figura 2). También podemos especificar si los accesos a través del puerto permitirán acceder a toda la red o sólo al servidor que recibe la llamada.

CONFIGURAR LOS PUERTOS RAS

En la configuración del RAS (figura 3), además de la configuración de los protocolos para hacer y recibir llamadas, podemos configurar la configuración de la codificación:

- *Permitir cualquier tipo de autentificación, incluso texto simple.* Al marcar esta opción, se permite a los clientes que usen este servicio utilizar la autentificación que ellos soliciten (*MS-CHAP*, *MD5-CHAP*, *SPAP* o *PAP*).

- *Exigir autentificación cifrada.* Al marcar esta opción, se permitirá a los clientes que usen este servicio utilizar la autentificación que ellos soliciten menos *PAP*; esto es, se requerirá contraseña cifrada de todos los clientes.

- *Exigir autentificación cifrada de Microsoft.* Al marcar esta acción, sólo se permitirá la conexión a los clientes que utilizan la autentificación *MS-CHAP*. También podemos marcar exigir datos cifrados para asegurarnos de que todos los datos enviados por la línea vayan codificados. *RAS* ofrece cifrado de datos mediante el aforismo RSA Data Security Incorporated RC4. Esta opción es la más segura de las tres.

- *Activar enlace múltiple.* Al marcar esta opción, se permite aumentar el ancho de banda de la transmisión al combinar múltiples vínculos físicos en un grupo; aunque sólo será efectivo cuando, tanto los clientes como los servidores *RAS,* tengan activado el enlace múltiple.

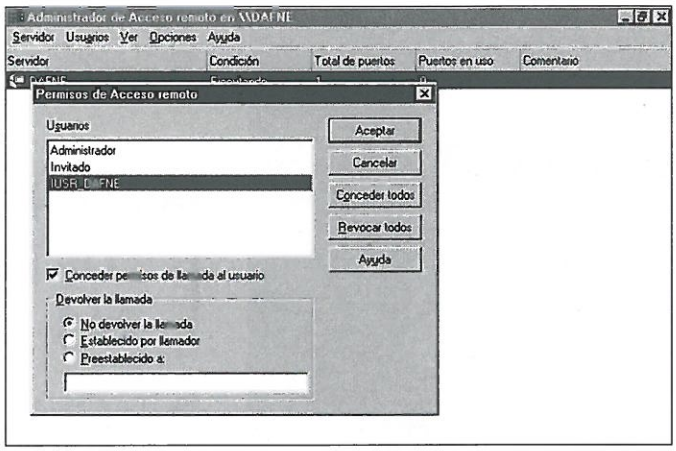


Figura 4

CONFIGURAR LAS CUENTAS DE USUARIO DE RAS

Desde el Administrador de acceso remoto, abriremos el menú Usuarios y seleccionaremos Permisos, lo cual nos mostrará la pantalla de la figura 4. Los permisos que podemos establecer son:

- *Conceder permisos de llamada al usuario*. Pese a la traducción que se le ha dado, significa que al marcar esta opción se permitirá al usuario hacer una llamada a este servidor RAS.

En el bloque devolver la llamada, se encuentran las siguientes opciones:

- *No devolver la llamada*. Al marcar esta opción, se desactiva la devolución de llamada al usuario que llama (es la opción por defecto).
- *Establecido por llamador*. Al marcar esta opción se pedirá al usuario que llama un número de teléfono para la devolución de llamada.
- *Preestablecido a*. Al marcar esta opción, cuando un usuario llame, se devolverá la llamada únicamente al número indicado (que deberemos escribir nosotros en el recuadro de debajo).

Si seleccionamos un usuario y hacemos modificaciones en las acciones indicadas anteriormente, se asignarán a dicho usuario. Pero, si marcamos *Conceder todos*, daremos dichos permisos a todos los usuarios y, si marcamos *Revocar todos*, eliminaremos los permisos de todos los usuarios.

EL MARCADO AUTOMÁTICO

El marcado automático asigna y mantiene las direcciones de red a las entradas de la guía telefónica de *RAS*, permitiendo que se marquen automáticamente cuando se hace referencia a ellas, ya sea desde una aplicación o desde la línea de comandos. Una dirección de red puede ser un nombre de *host* Internet, una dirección IP o un nombre de servidor *NetBIOS*.

El marcado automático también almacena cualquier conexión que se haya realizado en un vínculo *RAS* para su posible reconexión automática posterior. El marcado automático actúa de dos maneras:

- Si no está conectado a una red, el marcado automático intenta crear una conexión de red siempre que una aplicación haga referencia a una dirección remota.
- Si está conectado a una red, el marcado automático intenta crear una conexión de red solamente para las direcciones que ha aprendido anteriormente. De este modo, si se escribe incorrectamente el nombre de un servidor o de un *host* Internet no se producirá un intento de marcado automático.

Aunque el marcado automático se activa automáticamente cuando inicia el equipo, se puede desactivar si se desea.

No debemos confundir ni intentar que el marcado automático haga las veces de servidor *proxy*, ya que como *proxy* resulta muy malo. El marcado automático es útil sólo cuando deseamos acceder sólo a unas cuantas direcciones o establecer un acceso a redes externas muy restrictivo.

Si lo que deseamos es dar salida a toda la red con una única conexión a Internet, lo más recomendable es instalar un servidor *proxy* especializado. Además, este tipo de servidor proporcionará un espacio de caché que disminuirá nuestra factura telefónica, al gestionar de una forma más eficiente los contenidos solicitados por los usuarios.

EL PROTOCOLO PPTP

El protocolo *PPTP* (*Point to Point Tunneling Protocol*, Protocolo túnel punto a punto) proporciona la posibilidad de crear redes privadas virtuales; esto es, utilizar la infraestructura de Internet y enviar nuestros datos codificados a través de esta red. Es decir, que podemos crear una *WAN* de alcance mundial por un precio irrisorio.

PPTP soporta redes privadas virtuales multiprotocolo, permitiendo a los usuarios remotos acceder de forma segura a redes corporativas a través de Internet. De este modo podemos cambiar la carga de hardware, como modems y tarjetas *RDSI* desde el servidor NT a un procesador *front-end*, de modo que accedemos a nues-

tra red a través de un proveedor de servicios de Internet. Además, funciona con los protocolos *IP, IPX y NetBEUI*.

El protocolo *PPTP* trata nuestra red como si fuera una red *PSTN, RDSI* o *X25*, esto es, una *WAN* virtual.

En una *WAN de verdad*, un cliente remoto establece una conexión *PPP* con un servidor *RAS* sobre una red conmutada. Una vez que se ha realizado la conexión, se envían los paquetes *PPP* sobre la conexión conmutada hacia los servidores *RAS*, que a su vez lo enrutarán a la *LAN* de destino.

Esto mismo podemos conseguirlo llamando a un Proveedor de Servicios de Internet. El cliente hace primero una llamada al Proveedor de servicios, después de establecer esa conexión, el cliente hace otra conexión al servidor *RAS* que establece el "túnel" *PPTP* (figura 5). Un cliente que tenga *PPTP* como controlador de *WAN* puede acceder a los recursos de una LAN remota conectándose a un servidor *RAS* a través de Internet.

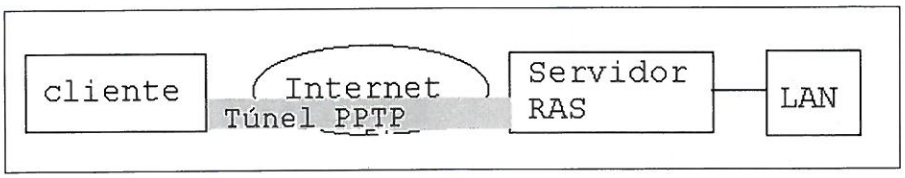


Figura 5.

SEGURIDAD

Los datos enviados a través del túnel *PPTP* se encapsulan en paquetes *PPP* y los datos se cifran. *RAS* permite cifrar grandes cantidades de datos usando RSA RC4 y una clave de sesión de 40 bits que se negocia en el momento de la conexión *PPP* entre el cliente *RAS* y el servidor *RAS* de Windows NT. Además de utilizar los vínculos *PPP* cifrados a través de Internet, una solución basada en *PPTP* permite también a Internet convertirse en una red principal para transportar el tráfico de acceso remoto *IPX* y *NetBEUI*. *PPTP* puede transferir el tráfico *IPX* porque encapsula y cifra los paquetes *PPP* de forma que pueden utilizar *TCP/IP*. Por tanto, la solución no depende sólo de las *LAN TCP/IP*.

INSTALACIÓN DE PPTP

Para instalar el protocolo PPTP haremos doble clic en el icono **Red** del *Panel de control*, y luego elegiremos la ficha *Protocolos*. Pulsamos sobre **Agregar** y seleccionamos el protocolo **Point to Point Tunneling Protocol** (figura 6). Luego nos pedirá el número de conexiones que queremos tener disponibles para *PPTP*, es decir, el número de redes privadas virtuales. Tras la copia de archivos, tendremos que seleccionar el puerto sobre el que deseamos instalar el protocolo *PPTP*. Otro punto muy importante consiste en activar el protocolo *PPTP* en cada uno de los adaptadores de la red empleada. Otra posibilidad interesante es el filtrado *PPTP*. Al seleccionar esta opción, se desactiva el adaptador de red seleccionado para el resto de los protocolos. De este modo, sólo se permitirá el acceso a los paquetes *PPTP*.

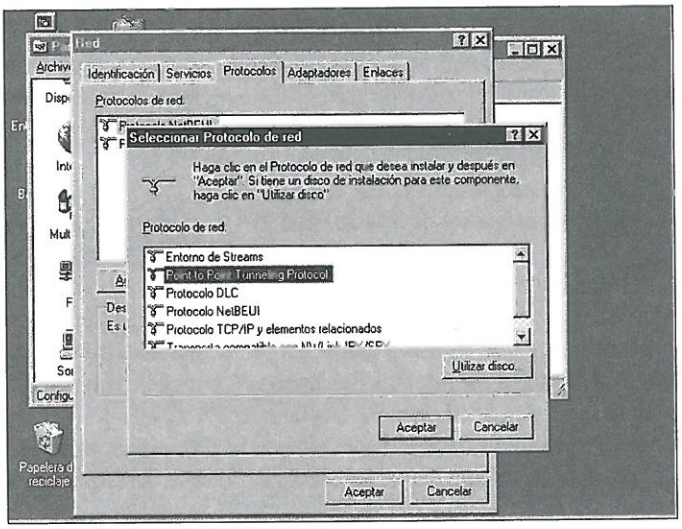


Figura 6.

Esto puede resultar útil cuando se tiene un sistema multitarjeta con una tarjeta de red con el filtrado *PPTP* activado que está conectada a Internet y otra tarjeta de red conectada a la red privada. Los clientes pueden usar *PPTP* para conectarse al equipo desde cualquier punto de Internet y tener acceso a la red privada. Por tanto, el único tráfico que puede tener acceso a la red privada son los paquetes *PPTP* de los clientes que se hayan autentificado mediante la autentificación de *RAS*. Así pues, pese a estar conectados a Internet, nadie que no esté autorizado en nuestra red tendrá acceso a ella.

Para activar el filtrado PPTP haremos doble clic en el icono **Red** y elegiremos la ficha **Protocolos**. Marcaremos el protocolo **TCP/IP** y haremos clic en **Propiedades**. A continuación elegiremos la opción **Avanzadas** en la ficha **Dirección IP**.

En el cuadro *Adaptador*, seleccionamos el adaptador de red para el que se desea especificar el filtrado *PPTP*. Los valores del filtrado *PPTP* de este cuadro de diálogo se definen sólo para la tarjeta de red seleccionada.

Finalmente, activamos el filtrado pulsando sobre **Activar filtro PPTP**.

LINUX

Bases de datos relacionales con PostGres

Juan Manuel Menéndez (jfrias@ ibm.net)

PostGres es un sistema de base de datos relacional que une las estructuras clásicas de estos sistemas con los conceptos de programación orientada a objetos, lo que convierte a Postgres en una base objeto-relacional. Permite la gestión de blobs (binary large object), que pueden almacenar cualquier contenido.

INSTALACIÓN

PostGres está siendo distribuida en varios formatos. En la distribución de RedHat viene en formato rpm, con tres paquetes para instalar: gestor de base de datos, bibliotecas de desarrollo y base de datos de ejemplo.

La versión incluida en RedHat 5 es la 6.2.1-7, que viene con los siguientes paquetes:

- postgresql-6.2.1-7.i386.rpm
- postgresql-data-6.2.1-7.i386.rpm
- postgresql-devel-6.2.1-7.i386.rpm

La instalación de dichos paquetes se realiza mediante el programa **Glint** para aquellos usuarios que dispongan de entornos X, aunque este entorno no es necesario para la ejecución de PostGres.

Aquellos usuarios que no estén aún en entorno gráfico podrán realizar la instalación mediante el comando rpm:

```
rpm –i postgresql-6.2.1-7.i386
rpm –i postgresql-data-6.2.1-7.i386
rpm –i postgresql-devel-6.2.1-7.i386
```

Siguiendo la línea de este esquema, la instalación del paquete gestor de base de datos que vamos a tratar carga en el sistema los ejecutables que aparecen en el cuadro 2, además de una completísima documentación imprimible y las páginas de manuales accesibles vía **man**.

Otro paso que realiza la instalación del gestor de bases de datos es la creación del usuario PostGres, que es el único que dispone de accesos a nivel del sistema operativo a las bases de datos.

La instalación de las bibliotecas de desarrollo lleva los ficheros *include* al directorio */usr/include/pqsql* y la biblioteca de funciones **libpq** a */usr/lib*.

El paquete de datos de ejemplo crea una base de datos en */var/lib/pqsql*.

Con el comando **rpm –q –p <paquere> -l**, se puede conseguir un listado de todo lo que se ha instalado.

Una vez realizada la instalación, en el siguiente arranque del sistema el gestor de PostGres se arrancará automáticamente, pero antes de abordar el funcionamiento con PostGres, resultará muy útil explicar ciertos conceptos sobre la arquitectura de este sistema gestor de base de datos.

ARQUITECTURA

Postgres utiliza el modelo cliente/servidor de un proceso por usuario. Este modelo consta de:

- Un programa supervisor.
- Uno o varios programas de aplicación.

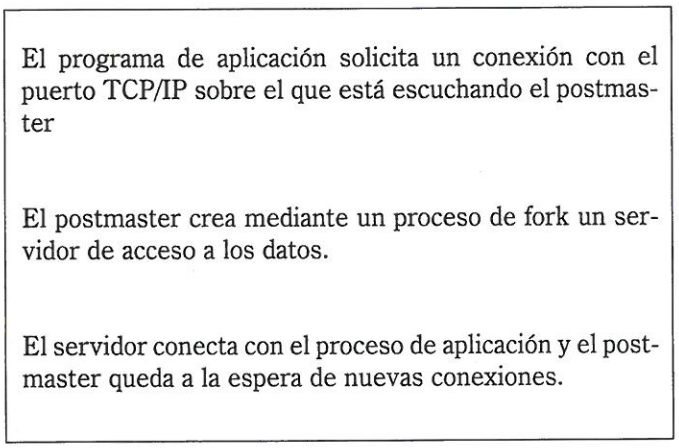

El programa de aplicación solicita un conexión con el puerto TCP/IP sobre el que está escuchando el postmaster
El postmaster crea mediante un proceso de fork un servidor de acceso a los datos.
El servidor conecta con el proceso de aplicación y el postmaster queda a la espera de nuevas conexiones.

Figura 1. Arquitectura de procesos de PostGres. Este proceso se realiza una vez por cada conexión.

- Un servidor de base de datos por cada programa de aplicación.

El programa supervisor se ejecuta en el sistema como un demonio y gestiona una colección de bases de datos.

Los programas de aplicación realizan las peticiones de acceso a una base de datos por medio de la biblioteca de funciones **libpq**.

La conexión entre el programa de aplicación y el servidor de base de datos se realiza por medio de un *socket TCP/IP*.

Con el comando rpm –q –p <paquere> -l, se puede listar todo lo que se ha instalado

El mecanismo de conexión es el siguiente:

El demonio supervisor, que en adelante llamaremos **postmaster**, permanece a la escucha en un puerto *TCP/IP*. Cuando un cliente desea acceder a la base de datos lanza un **connect** dirigido al puerto del *postmaster*, el cual al recibir la petición realiza un **fork** de sí mismo. El nuevo proceso creado con la función *fork* permanece en sesión con la aplicación cliente, mientras el *postmaster* vuelve a la escucha para atender otras peticiones de conexión.

Este modelo de arquitectura, al estar apoyado en *TCP/IP*, permite que los clientes puedan estar tanto en local como en remoto, utilizando en local la dirección **loopback 127.0.0.1**.

Con respecto a la arquitectura de los datos, las bases de datos se almacenan en directorios de tipo standard. Cuando se arranca el *postmaster*, se le indica en qué directorio van a estar las bases de datos. La estructura de estos directorios parte de la variable **PGDDATA** que puede ser suministrada de varias formas:

- Por medio de variables de entorno.
- En la sentencia de arranque con el parámetro **-D**.

A partir de este punto la estructura de directorios queda de la siguiente forma:

Ficheros denominados con el nombre **pg_xxxxxxx**. Conteniendo varias tablas que utiliza el propio sistema para gestionarse.

Directorio base, que contiene los bases de datos, estando representada cada una de ellas por un directorio cuyo nombre coincide con el de la base de datos.

Dentro de cada directorio que almacena las diversas bases de datos, existen una serie de ficheros, una parte de los cuales tiene el nombre con la forma **pg_xxxxxx**, que contiene las tablas de sistema que gestionan la base de datos, a la que pertenecen, y una serie de ficheros cada uno de los cuales coincide con las clases (tablas), que están definidas en dicha base de datos y que contienen los datos propiamente dichos.

Cuadro 1. Paquetes que se incluyen con la distribución RedHat 5.0

postgresql-6.2.1-7.i386.rpm
postgresql-data-6.2.1-7.i386.rpm
postgresql-devel-6.2.1-7.i386.rpm

CONCEPTOS DE BASE DE DATOS

Los conceptos clásicos de las bases de datos relacionales se unen a los conceptos de programación orientada a objetos. En PostGres las tablas se denominan clases, las filas se denominan instancias y las columnas se denominan atributos.

Un concepto que aparece en PostGres y lo distingue de otros modelos es la Herencia. Este nuevo concepto viene claramente de la POO, y se ajusta a ésta perfectamente. Así pues, cuando se crea una nueva clase heredada de otra, la clase creada adquiere todos las características de la clase de la que proviene, más las características que se definan en la nueva clase.

De la misma forma cuando se hereda una clase, se está creando una nueva tabla igual a la origen de la herencia.

Poniendo un ejemplo si se tiene la tabla tabla-1 que se ha creado con la siguiente sentencia SQL, dando como resultado:

```
CREATE TABLE tabla-1 (
        Campo1 text
        Campo2 int
        )
```

Cuadro 2. Ficheros que se instalan con el paquete PostGres.

```
/etc/rc.d/init.d/postgresql
/usr/bin/cleardbdir
/usr/bin/createdb
/usr/bin/createuser
/usr/bin/destroydb
/usr/bin/destroyuser
/usr/bin/initdb
/usr/bin/pg_dump
/usr/bin/pg_dumpall
/usr/bin/pg_id
/usr/bin/pg_passwd
/usr/bin/pg_version
/usr/bin/postgres
/usr/bin/postmaster
/usr/bin/psql
```

A partir de este momento nos encontramos con que se crea la tabla tabla-2 con la siguiente sentencia SQL, a continuación definida:

```
CREATE TABLE tabla-2 (
        Campo3 int)
INHERITS (tabla-1).
```

Como resultado de esta sentencia SQL la nueva tabla tendrá los atributos: Campo1, Campo2, Campo3 .

Por lo demás, al seguir PostGres el modelo relacional de base datos, los conocimientos standard de este modelo se pueden aplicar para empezar a trabajar con este sistema.

ARRANQUE DE POSTGRES

Como se mencionó anteriormente, el demonio *postmaster* debe estar arrancado para poder tener acceso a las bases de datos. La instalación de PostGres crea un enlace de nombre **S85postgresql** en el directorio */etc/rc.d/rc3.d/* que apunta al fichero */etc/rc.d/init.d/postgresql*. Este enlace es llamado por el sistema de la forma:

```
/etc/rc.d/rc3.d/S85postgresql start
```

durante el arranque del sistema, cuando éste pasa a modo multiusuario.

Cuadro 3. Ficheros que se descargan con el paquete PostGress-data.

```
/var/lib/pgsql/PG_VERSION
/var/lib/pgsql/base
/var/lib/pgsql/base/template1
/var/lib/pgsql/base/template1/PG_VERSION
/* Ficheros que componen la estructura de ba base template1
/var/lib/pgsql/base/template1/pg_XXXXXXXXX
/* Ficheros que componen la estructura del motor de acceso a datos
/var/lib/pgsql/pg_database
/var/lib/pgsql/pg_defaults
/var/lib/pgsql/pg_demon
/var/lib/pgsql/pg_geqo.sample
/var/lib/pgsql/pg_group
/var/lib/pgsql/pg_hba.conf
/var/lib/pgsql/pg_hosts
/var/lib/pgsql/pg_log
/var/lib/pgsql/pg_magic
/var/lib/pgsql/pg_server
/var/lib/pgsql/pg_time
/var/lib/pgsql/pg_user
/var/lib/pgsql/pg_variable
```

Se puede comprobar que este demonio está arrancado con los siguientes comandos:

```
ps –x | grep postmaster | grep –v grep
```

y:

```
netstat –an | grep 5432
```

El primer comando permite comprobar que existe un proceso llamado *postmaster* arrancado en el sistema, y el segundo comando que existe un programa escuchando (*LISTEN*), en el puerto **5432**.

Una vez se ha realizado la instalación, en el siguiente arranque del sistema el gestor de PostGres se arrancará automáticamente

Como ya se ha dicho, las bases de datos están ubicadas como subdirectorios dentro del directorio *$PGDATA/base*. En el procedimiento de arranque *postgresql* se puede observar la sentencia de arranque completa del *postmaster*:

```
Postmaster –S –D /var/lib/sqlpg
```

Salvo para empezar a probar, no es aconsejable utilizar directorios del sistema para almacenar datos de usuario, por lo que lo primero que se hará es modificar la sentencia de arranque cambiando el directorio del sistema por uno de usuario.

Otro valor que puede necesitar ser cambiado es el número del puerto *TCP/IP*. Por defecto, el *postmaster* asume el valor 5420 pero se puede indicar cualquier otro por medio del parámetro **–P**.

Existen otros parámetros que afectan al modo en que se arranca *postmaster*.

La forma normal de arranque de cualquier proceso UNIX se apropia de la pantalla y del teclado desde donde dicho proceso ha sido arrancado. Esta forma de arrancar no es muy adecuada para procesos de larga duración. Para evitar esto, este tipo de procesos se lanzan en *background*. Para lanzar un proceso de esta manera lo único que es necesario hacer es colocar el signo ampersand (&) al final del comando de arranque del proceso:

```
NombreProceso param1 param2 param3 ..... &
```

Un proceso lanzado en *background* deja liberado el teclado, pero no la unidad de salida standard del proceso, que es el monitor; para dejar independiente tanto la unidad de entrada como la de salida es necesario redirigir la salida hacia un fichero:

```
NombreProceso param1 param2 param3 ..... > fichero.salida &
```

El programa supervisor se ejecuta en el sistema como un demonio y gestiona una colección de bases de datos

Ya tenemos el proceso en *background* totalmente independiente en sus dispositivos de entrada y salida, pero aún queda una dependencia extremadamente fuerte que es la relación del proceso con su proceso padre que corresponde con la sesión de *shell* del usuario que ha arrancado el proceso. Es decir la relación que existe entre cada proceso y el proceso padre. En este caso si el usuario que ha arrancado el proceso que está en *background* sale del sistema, todos los procesos hijos del mismo acaban. Para independizar los procesos hijos del proceso padre existen dos técnicas que explicaremos a continuación: la primera ellas es arrancar el proceso con el comando **nohup**:

```
nohup NombreProceso param1 param2 param3 ..... &
```

La otra forma de conseguir que un proceso sea independiente tanto de los dispositivos de entrada y salida como de proceso padre se realiza durante el diseño del proceso, mediante una serie de instrucciones que se realizan al iniciarlo, y es independiente de la forma de arranque del mismo desde la *shell*. Si se arranca *postmaster* con el parámetro **-S**, se consigue que el proceso vaya directamente a *background*, desligado del teclado y monitor e independiente del proceso padre.

La independencia de un proceso con respecto al proceso padre es relativa, pues todo proceso que esté arrancado en el sistema debe tener un proceso padre, a excepción del proceso **init**. Este proceso que tiene una identificación de (*PID*) de 1, es el único que está permanentemente arrancado, y para independizar un proceso de su proceso padre lo que hay que hacer es poner como padre del proceso independizado al proceso **init**.

TRABAJANDO CON POSTGRES

A partir de este momento existen diversas maneras de trabajar con PostGres que vamos a ver a lo largo del artículo. La primera de ellas es mediante la línea de comandos. Con la línea de comandos se puede trabajar interactivamente con las bases de datos y realizar tareas de administrador de la misma, para después lograr mediante interfaz de programación se pueden realizar también las mismas funciones que mediante las lí-

Cuadro 4. Ficheros que se descargan con el paquete PostGres-Devel.

```
/usr/include/pgsql
/usr/include/pgsql/access
/usr/include/pgsql/access/attnum.h
/usr/include/pgsql/c.h
/usr/include/pgsql/config.h
/usr/include/pgsql/fmgr.h
/usr/include/pgsql/include
/usr/include/pgsql/lib
/usr/include/pgsql/lib/dllist.h
/usr/include/pgsql/libpq
/usr/include/pgsql/libpq-fe.h
/usr/include/pgsql/libpq/libpq-fs.h
/usr/include/pgsql/libpq/pqcomm.h
/usr/include/pgsql/os.h
/usr/include/pgsql/port
/usr/include/pgsql/port/linux
/usr/include/pgsql/postgres.h
/usr/include/pgsql/postgres_ext.h
/usr/include/pgsql/utils
/usr/include/pgsql/utils/elog.h
/usr/include/pgsql/utils/geo_decls.h
/usr/include/pgsql/utils/palloc.h
/usr/lib/libpq.a
/usr/lib/libpq.so
/usr/man/man3/catalogs.3
/usr/man/man3/large_objects.3
/usr/man/man3/libpq.3
/usr/man/man3/oracle_compat.3
/usr/man/man3/pgbuiltin.3
```

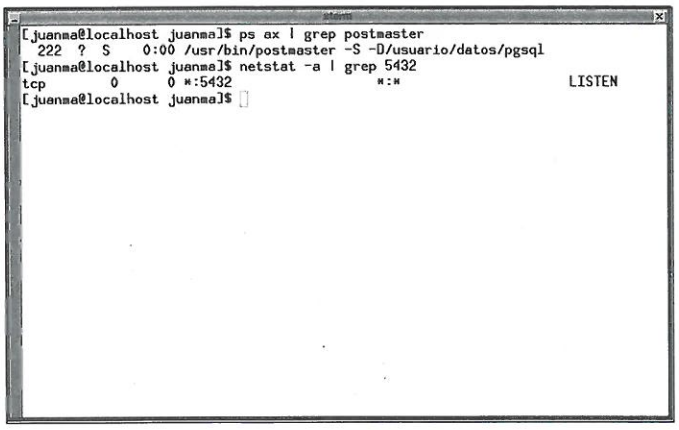

[juanma@localhost juanma]$ ps ax | grep postmaster
222 ? S 0:00 /usr/bin/postmaster -S -D/usuario/datos/pgsql
[juanma@localhost juanma]$ netstat -a | grep 5432
tcp 0 0 *:5432 *:* LISTEN
[juanma@localhost juanma]$

Figura 2. La salida da los comandos ps y netstat, indican que el postmaster está correctamente arrancado.

neas de comandos y se utiliza para el desarrollo de aplicaciones.

Se accede a la interfaz de comandos mediante la sentencia **psql** lanzada desde el *prompt* del *shell*. Si se ejecuta sin parámetros, el proceso intenta conectarse a una base datos con el mismo nombre del usuario y si ésta no existe, lo que será normal si es la primera vez que se accede, dará un error.

El nuevo proceso creado con la función fork permanece en sesión con la aplicación cliente, mientras el postmaster vuelve a la escucha para atender a otras peticiones de conexión

En este punto es necesario hacer hincapié en que es muy conveniente crearse un usuario para acceder a PostGres, en vez de hacerlo con *root*.

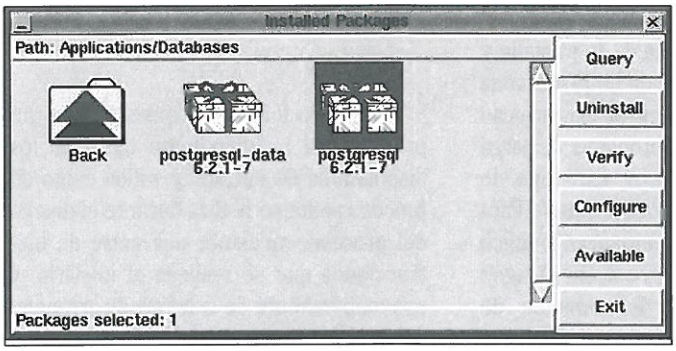


Figura 3.
Los paquetes de PostGres pueden ser instalados desde el entorno X-Windows con el programa glint.

Si se ha instalado el paquete **postgresql-data**, se habrá creado una base de datos llamada **template1**. Para arrancar con esa base de datos se entrará en el entorno de comando de la siguiente forma:

```
psql template1
```

En el entorno de comandos existen dos tipos de comandos: los comandos de *backslash* y los comandos de SQL. Los primeros permiten trabajar con el entorno, mientras que los segundos son las sentencias de gestión de bases de datos tales como: **create table**, **insert**, **select**, **update**, **delete**, etc.

Los comandos de *backslash* se denominan así porque deben de comenzar con dicho carácter (\) para poder ser empleados.

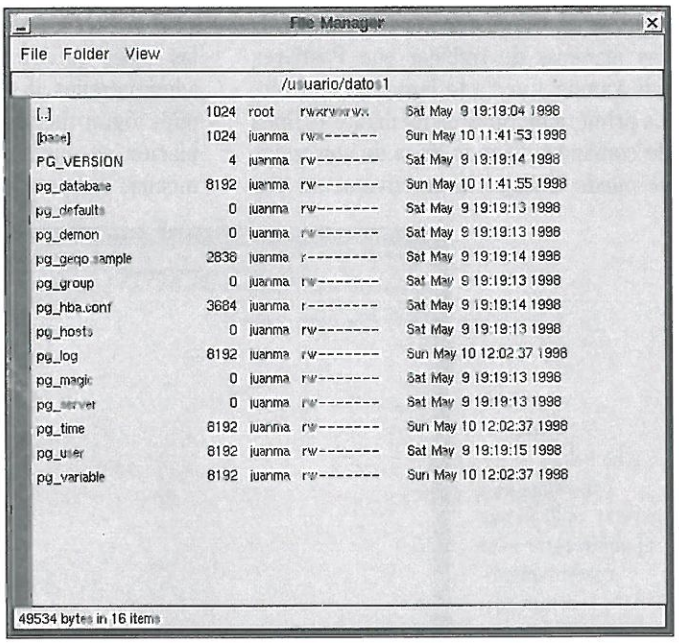

File Manager
File Folder View
/usuario/dato1
[..] 1024 root Sat May 9 19:19:04 1998
[base] 1024 juanma Sun May 10 11:41:53 1998
PG_VERSION 4 juanma Sat May 9 19:19:14 1998
pg_database 8192 juanma Sun May 10 11:41:55 1998
pg_default 0 juanma Sat May 9 19:19:13 1998
pg_demon 0 juanma Sat May 9 19:19:13 1998
pg_geqo 2838 juanma Sat May 9 19:19:14 1998
pg_group 0 juanma Sat May 9 19:19:13 1998
pg_hba.conf 3684 juanma Sat May 9 19:19:14 1998
pg_hosts 0 juanma Sat May 9 19:19:13 1998
pg_log 8192 juanma Sun May 10 12:02:37 1998
pg_magic 0 juanma Sat May 9 19:19:13 1998
pg_time 8192 juanma Sun May 10 12:02:37 1998
pg_user 8192 juanma Sat May 9 19:19:15 1998
pg_variable 8192 juanma Sun May 10 12:02:37 1998
49534 bytes in 16 items

Figura 4.
En el directorio de datos están todas las tablas de sistemas más las de datos en el subdirectorio base.

Para acceder a las ayudas de ambos tipos de entornos hay que teclear:

\h : Para obtener la ayuda de los comandos SQL

\? : Para obtener la ayuda de los comandos de *backslash*.

Para trabajar con este entorno hay que tener en cuenta las siguientes reglas:

- Los comandos de backslash se ejecutan instantáneamente.
- Los comandos de SQL se almacenan en buffer y no se ejecutan hasta que se de la orden de ejecución.
- Es posible editar el buffer de comandos para realizar modificaciones.
- Los comandos de SQL se pueden guardar en ficheros y ser llamados para ejecutar desde el entorno de PostGres.

Entre los mandatos más usuales del entorno se encuentran:

- \e: Edición de buffer (con vi)
- \i < *fichero* > : lee una sentencias de SQL del fichero y las lleva al buffer.
- \g: ejecuta las sentencias de SQL almacenadas en el buffer.
- \d: muestra las bases de datos.
- \dt: muestra las tablas en una base de datos.
- \q: finaliza la sesión PostGres

Las sentencias de SQL se van introduciendo en el buffer de trabajo con la tecla **Enter**. Una vez que se ha creado la sentencia SQL completa, para ejecutarla hay dos opciones: ejecutar el comando \g,

Cuadro 5. Procedimiento de arranque de postmaster. En negrita la sentencia de arranque de PostGres. En el parámetro -D se ha especificado el nuevo directorio de datos.

```
#! /bin/sh
# chkconfig: 345 85 15
# description: Starts and stops the PostgreSQL
# Source function library.
. /etc/rc.d/init.d/functions
# Get config.
. /etc/sysconfig/network
# Check that networking is up.
# Pretty much need it for postmaster.
[ ${NETWORKING} = "no" ] && exit 0
 [ -f /usr/bin/postmaster ] || exit 0
# See how we were called.
case "$1" in
  start)
        echo -n "Starting postgresql service: "
        su postgres -c '/usr/bin/postmaster -S -D/usuario/datos1'
        sleep 1
        pid=`pidof postmaster`
        echo -n "postmaster [$pid]"
        touch /var/lock/subsys/postmaster
        echo
        ;;
  stop)
        echo -n "Stopping postgresql service: "
        killproc postmaster
        sleep 2
        rm -f /var/lock/subsys/postmaster
        echo
        ;;
  status)
        status postmaster
        ;;
  restart)
        $0 stop
        $0 start
        ;;
  *)
        echo "Usage: postgres.init {start|stop|status|restart}"
        exit 1
esac
exit 0
```

para proceder a la ejecución del buffer de SQL, o finalizar la sentencia con punto y coma, una vez se introduce el punto y la coma se pasa a ejecutar la sentencia SQL.

En el CD-ROM de la revista se adjuntan tres ficheros con sentencias SQL que permiten crear una base de datos llamada **bd1** con dos tablas: **tabla1** y **tabla2**. La segunda hereda características de la primera. Otro fichero inserta datos en las tablas y el tercer fichero lista su contenido, completando los anteriores.

Para ejecutar las sentencias de SQL hay que leer el fichero que se desea ejecutar **\i crea.sql**, una vez leído, podemos verificar su contenido con **\e**, y pasar a ejecutarlo con **\g**. Una vez ejecutado el SQL de creación podemos ver las nuevas tablas con los comandos **\d** y **\dt**.

Los conceptos clásicos de las bases de datos relacionales, se unen a los conceptos de programación orientada a objetos

Antes de acceder a la base datos, hay que crearla desde el entorno con el comando **createdb**. Una vez creada la base de datos, es necesario conectarse a ella para poder trabajar. Para conectarse a una base de datos hay que ejecutar el comando **psql <nombre base datos>**.

Para eliminar una base datos se utiliza el comando **destroydb**. Es importante sesaltar que estos comandos se ejecutan desde el *prompt* del sistema y no desde el entorno de PostGres.

PARADA Y OTRAS TAREAS CON POSTMASTER

Para parar el *postmaster* se puede usar el comando **kill**, evitando dar la opción **–9** o **–KILL**, con el fin de que los procesos puedan ir finalizando correctamente y evitar posibles corrupciones de datos.

Una de las tareas importantes que se debe realizar en una base de datos son

Cuadro 6. Listado del programa prog.c

```
#include <stdio.h>
#include <libpq-fe.h>
main()
{
char *pghost, *pgport, *pgoptions, *pgtty;
char * nombreDB;
int i, j, k;
PGconn* conn;
PGresult *res;
/* Fin de definicion de variables */
pghost = NULL;
pgport = NULL;
pgoptions = NULL;
pgtty = NULL;
nombreDB = "bd1";
conn = PQsetdb(pghost,pgport,pgoptions,pgtty,nombreDB);
if (PQstatus(conn) == CONNECTION_BAD)
{
        printf("Error en conexion a %s ",nombreDB);
        printf("%s",PQerrorMessage(conn));
        PQfinish(conn);
        exit(1);
}
printf("Conexion realizada con la base de datos: %s\n",PQdb(conn));
printf("Equipo: %s\n",PQhost(conn));
printf("Puerto de conexion: %s\n",PQport(conn));
printf("Terminal tty: %s\n",PQtty(conn));
printf("Opciones de conexion: %s\n",PQoptions(conn));
printf("\n");
res = PQexec(conn,"BEGIN");
if (PQresultStatus(res) != PGRES_COMMAND_OK)
{
        printf("Error en BEGIN");
        PQclear(res);
        PQfinish(conn);
        exit(1);
}
PQclear(res);
res = PQexec(conn,"DECLARE cursor_1 CURSOR FOR select * from tabla1");
if (PQresultStatus(res) != PGRES_COMMAND_OK)
{
        printf("Error en DECLARE CURSOR");
        PQclear(res);
        PQfinish(conn);
        exit(1);
}
PQclear(res);
res = PQexec(conn,"FETCH all in cursor_1");
if (PQresultStatus(res) != PGRES_TUPLES_OK)
{
        printf("Error en FETCH");
        PQclear(res);
        PQfinish(conn);
        exit(1);
}
k = PQnfields(res);
for ( i = 0; i < k; i++)
{
        printf("%-12s",PQfname(res,i));
}
printf("\n");
for ( i = 0; i < PQntuples(res); i++)
{
        for (j = 0; j < k; j++)
        {
                printf("%-12s",PQgetvalue(res,i,j));
        }
        printf("\n");
}
PQclear(res);
res = PQexec(conn,"CLOSE cursor_1");
PQclear(res);
res = PQexec(conn,"END");
PQclear(res);
PQfinish(conn);

exit(0);
}
```

las copias de seguridad. Las copias de seguridad de las bases de datos Postgres se puede realizar bien con el comando **dump** o mediante el comando **tar.**

Si el *postmaster* tiene una finalización anormal, al volver a arrancarlo se pueden producir errores en la asignación de memoria compartida y semáforos. Si se da esta situación es necesario ejecutar el comando **ipcclean** que restaura todos los recursos *ipc* que existan en el sistema. Si se tienen varias instancias del *postmaster* ejecutando, es necesario pararlas todas, pues el programa **ipcclean** elimina todos los recursos *ipc* de sistema cuyo propietario sea el usuario postgres.

En PostGres, de la misma forma, cuando se hereda una clase se está creando una nueva tabla igual a la de origen de la herencia

La creación de directorios para las bases de datos se realiza por medio del comando **initdb:**

```
initdb –r /usuario/datos1 –l /usr/lib
```

r:directorio donde se van a ubicar las bases de datos.

l: directorio de plantillas que en la instalación se crean en */usr/lib*.

Para mover una base de datos de una ubicación a otra distinta primero se para el *postmaster* y a continuación se ejecuta el comando:

```
cp –rp /directorio/viejo /directorio/nuevo
```

Una vez realizada la copia se rearranca el *postmaster* indicando cuál es el nuevo directorio mediante el parámetro **–D** o la variable de entormo **PGDATA.**

Cuadro 7. Estados que puede presentar una conexión y una query.

Estado de las conexiones(Obtenido por PQstatus):
PG_CONNECTION_OK
PG_CONNECTION_BAD

Estado de las querys (Obtenido por PQresultStatus):
PGRES_EMPTY_QUERY
PGRES_COMMAND_OK
PGRES_TUPLES_OK
PGRES_COPY_OUT
PGRES_COPY_IN
PGRES_BAD_RESPONSE
PGRES_NONFATAL_ERROR
PGRES_FATAL_ERROR

Cuando una base datos tiene un alto índice de volatibilidad, es decir cuando constantemente se están modificando datos, las estadísticas que mantiene el sistema sobre las bases de datos quedan desactualizadas. Estas estadísticas son utilizadas por el gestor de bases de datos para evaluar las estrategias de acceso y tomar decisiones que afectan al rendimiento de la base de datos. Las estadísticas del sistema no se actualizan *on-line* pues son un proceso pesado para el sistema, si no que se actualizan periódicamente mediante el comando **vacuum**.

Un buen criterio en bases de datos que tienen aplicaciones reales es realizar el proceso de actualización de estadisticas una vez al día, en horas fuera de servicio de la base de datos, para que este proceso no afecte negativamente el rendimiento del sistema.

GRANDES OBJETOS

Como se ha mencionado anteriormente, PostGres permite almacenar además de datos convencionales, datos binarios tales como imágenes, sonidos, programas ejecutables, etc. La creación de tablas que contienen atributos que son grandes objetos se realiza por medio del tipo de datos *oid*. En el fichero **creaimag.sql** está la sentencia SQL que crea una tabla uno de cuyos campos es un gran objeto.

```
CREATE TABLE  ( texto text, grafico oid)
```

Para manejar desde SQL los objetos de este tipo PostGres suministra dos funciones, una de ella permite importar ficheros a objetos y la otra permite exportar objetos a ficheros.

Se accede a la interfaz de comandos mediante la sentencia psql lanzada desde el prompt del shell

Para cargar ficheros en la base de datos se usa la función **lo_import** embebida en la sentencia **INSERT** de SQL:

```
INSERT INTO imágenes (texto, grafico)
        VALUES('Texto.....',lo_import('nombre.del.fichero')
```

Para extraer ficheros de la base de datos se usa la función **lo_export** igualmente embebida en la sentencia SQL:

```
SELECT lo_export(imágenes.grafico,
        'nombre.del.fichero')
        FROM imágenes WHERE texto =
        'Texto.....'
```

FUNCIONES BAJO SQL

La sintaxis de las sentencias anteriores hacen referencia a una función en lugar de a un campo de la tabla. PostGres permite la creación de funciones que luego pueden ser invocadas desde SQL. Estas funciones pueden estar escritas tanto en lenguaje C como en lenguaje SQL. La funciones se crean con la sentencia **CREATE FUNCTION**, cuya sintaxis completa es:

```
CREATE FUNCTION nombreFuncion()
        RETURNS tipovariable
        AS ' definicion función ' LANGUAGE 'tipo
        lenguaje'
```

La forma de definir la función es distinta según sea el tipo de lenguaje, si el tipo de lenguaje es SQL la función se define en formato texto pero si el tipo de lenguaje es C en la definición de la función se pone el nombre de la biblioteca de funciones.

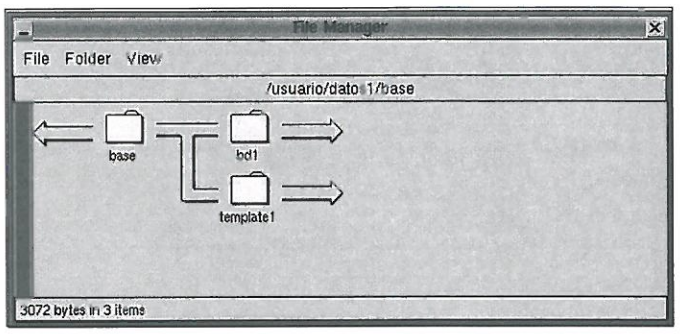


Figura 5.
El directorio base contiene un subdirectorio para cada base de datos. En estos directorios están contenidas las tablas de datos.

En el caso de una función en lenguaje C:

```
CREATE FUNCTION nombreFuncion(int)
    RETURNS tipovariable
    AS ' /usr/lib/funciones.so ' LANGUAGE 'c'.
```

Si la función es en lenguaje SQL a definición queda:

```
CREATE FUNCTION numdb() RETURNS int
    AS ' SELECT COUNT (*) FROM
    PG_DATABASE ' LANGUAGE 'SQL'.
```

Esta última función suministra el número de bases de datos que están gestionadas por un *postmaster*.

La llamada a esta función se realiza de la siguiente forma:

```
SELECT numxdb().
```

En cualquier caso la creación de funciones tanto en lenguaje C como SQL requiere por sí sola un artículo completo.

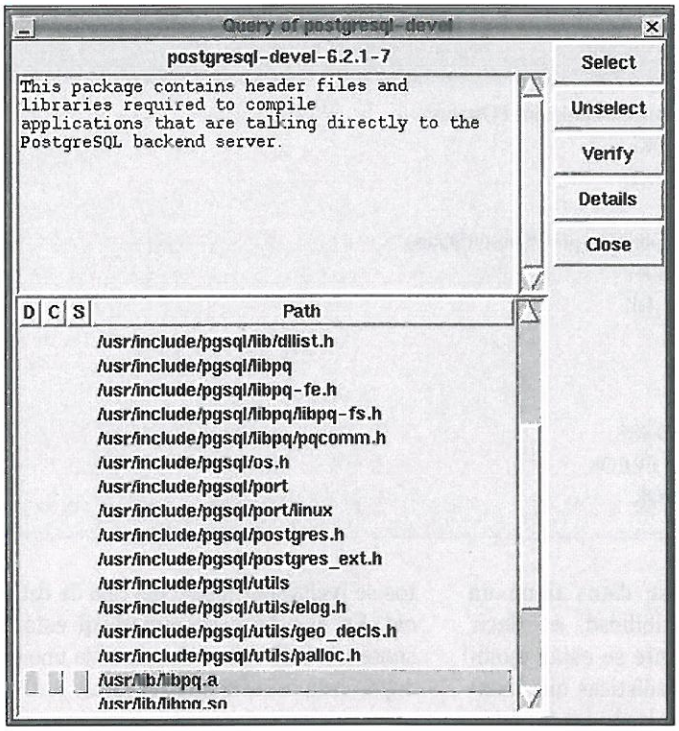


Figura 7.
Con glint se puede ver el detalle de los ficheros cargados con el paquete de desarrollo de postgres.

PROGRAMACIÓN EN C

PostGres dispone de un API de programación implementado en la biblioteca **libpq.a**. Esta biblioteca se instala en el paquete de desarrollo y tiene un juego de funciones que permite pasar las *queries* de SQL al servidor de PostGres y recibir los resultados de las mismas.

Antes de pasar a ver la programación con PostGres vamos a ver cómo hay que compilar y enlazar los programas con la biblioteca **libpq.a**. La biblioteca de funciones **libpq** se instala en el directorio */usr/lib* por que lo queda dentro del camino de busca standard del montador de enlace.

La directiva que necesita el montador de enlace es una referencia a dicha biblioteca con parámetro **–l**. EL valor que hay que suministrar a este parámetro es el nombre de la biblioteca quitando el prefijo **lib** y el sufijo **.a**, así pues la referencia a *libpq* queda de la forma **–lpq**, es importante indicar que no debe de existir ningún espacio en blanco entre el indicador de la directiva y su valor.

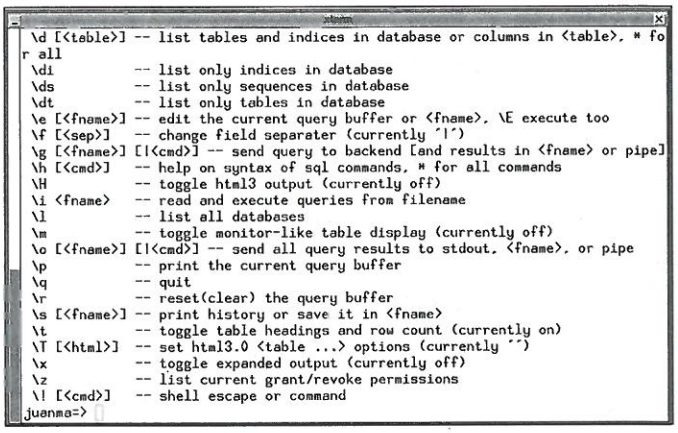


Figura 6.
El juego de comandos de backslash permite realizar tareas de gestión y acceso no planificado a las bases de datos.

Para lanzar un proceso en background lo único que es necesario hacer es colocar el signo ampersand (&) al final del comando de arranque del proceso

Los programas que usen *libpq* además deberán incluir el fichero de cabecera **libpq-fe.h** (*Front End* de *libpq*) que se instala también con el paquete de desarrollo de Postgres en el directorio /usr/include/pgsql, este directorio no se encuentra en el camino de búsqueda del compilador por lo que hay que indicárselo con la directiva **–I**.

Así pues una sentencia para compilar un programa con las funciones de

PostGress queda de la siguiente forma:

```
gcc –o nombre_objeto –I /usr/inclue/pgsql –lpq
      nombre_fuente.c
```

En el CD-ROM de la revista se incluye un fichero **Makefile** para compilar el programa de ejemplo que también se incluye en el CD-ROM, y que se comentará en las páginas siguientes.

En el entorno de comandos existen dos tipos de comandos: los comandos de backslash y los comandos de SQL

Para realizar la compilación y enlazado del programa simplemente hay que ejecutar la sentencia: **make**. Esta sentencia, si se ejecuta sin parámetros, busca el fichero **Makefile** y ejecuta las sentencias que contiene.

La forma llamar a las compilaciones es la siguiente:

- make prog (Para el programa prog).
- make prog1 (Para el programa prog1).
- make prog2 (Para el programa prog2).
- make todo (Para los tres programas).

Tal y como se ha visto anteriormente, al entrar en el entorno de comandos *psql* era necesario suministrar una serie de parámetros, como son el nombre del equipo donde se está ejecutando el servidor, el número de puerto de conexión y el nombre de la base datos. También es necesario suministrar estos parámetros al realizar una conexión desde del API, existiendo tres formas de hacerlo:

- Dejar los valores por defecto.
- Codificar estos valores en el propio programa.
- Suministrar dichos valores con las variables de entorno: **PGHOST, PGOPTIONS, PGPORT, PGTTY, PGDATABSE.**

La forma de suministrar los valores por medio de variables de entorno se realiza asigándolos y posteriormente exportando las variables en cuestión:

```
set PGPORT=5432; export PGPORT
```

Las variables exportadas pueden ser recuperadas desde un programa por medio de la función **getenv()**.

Cuadro 8. Algunas de las tablas que forman el catálogo del sistema.

pg_database	Bases de datos existentes
pg_class	Clases (tablas)
pg_attribute	Atributos (campos)
pg_index	Indices Secundarios

FUNCIONES DEL API

Las funciones suministradas por el API se pueden agrupar en:

- Funciones de gestión de la conexión.
- Funciones de ejecución de *Queries*.
- Funciones de depuración.

Los nombres de las funciones empiezan por las dos letras PQ. Entre las funciones de gestión de la conexión están:

- *PQsetdb*: Realiza la conexión a una base de datos.
- *PQfinish*: Cierra la conexión con una base de datos.
- *PQhost, PQdb, PQoption, PQport, PQtty*: devuelven los valores de los parámetros de conexión.
- *PQstatus*: Devuelve el estado de la conexión.
- *PQerrorMessage*: Devuelve el mensaje de error asociado con la conexión.

Las funciones de ejecución:

- *PQexec*: envía o solicita una *query* al servidor.
- *PQresultStatus*: Devuelve el resultado de la *query*.
- *PQntuples*: Devuelve el número de instancias (filas) de la *query*.
- *PQnfields*: Devuelve el número de atributos (campos) de la *query*.
- *PQfname*: Devuelve el nombre del atributo.
- *PQfnumber*: Devuelve el número de orden del atributo.
- *PQftype*: Devuelve el tipo asociado con el atributo.
- *PQfsize*: Devuelve el tamaño del atributo.
- *PQgetvalue*: Devuelve el valor del atributo.
- *PQclear*: Libera la memoria asociada al resultado de la *query*.

Los comandos de SQL se pueden guardar en ficheros y ser llamados para ejecutar desde el entorno de PostGres

Una vez presentadas estas funciones contaremos con dos estructuras dos estructuras que participan también en el diálogo entre el servidor y el programa de aplicación: la estructura **PGconn** y la estructura **PGresult**.

La primera se obtiene como resultado de la función **PQsetdb**, y es utilizada como parámetro en las demás funciones de gestión de la conexión. La segun-

suscríbete

a Sólo Programadores

y consigue un magnífico descuento

suscripción normal

ahorro **20%**

12 revistas (1 año) por sólo...

9.350 ptas.

suscripción estudiantes (carreras técnicas)

ahorro **40%**

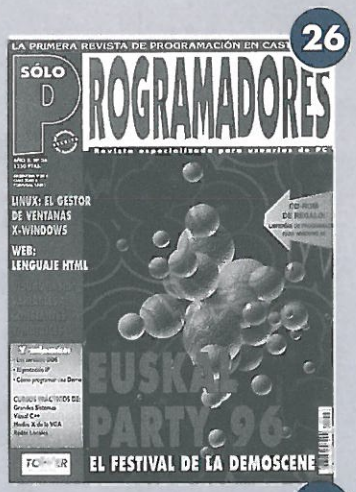
12 revistas (1 año) por sólo...

7.050 ptas

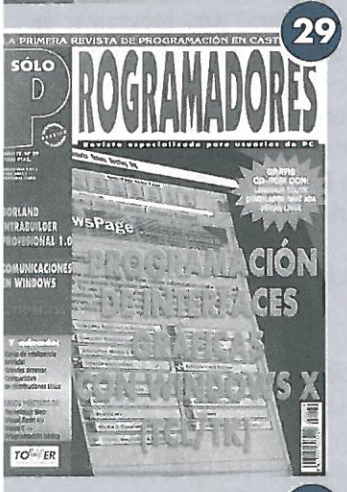
números atrasados

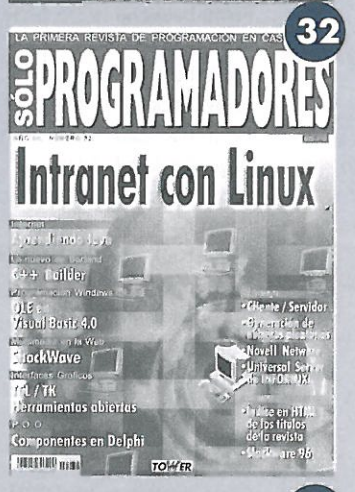
SÓLO PROGRAMADORES

completa ya tu colección



28
SÓLO PROGRAMADORES
ANALIZAMOS EL ESTADO DE LA INTELIGENCIA ARTIFICIAL
VISUAL C++ PROGRAMACIÓN DEL GDI
ESPECIAL HERRAMIENTAS RAD
DESARROLLO RÁPIDO DE APLICACIONES CON OPTIMACE





Cuadro 9 Sesión de trace del programa prog2.

```
To backend> QBEGIN
From backend> C
From backend> "BEGIN"
To backend> Q
From backend> I
From backend>
To backend> QDECLARE cursor_1 CURSOR FOR select * from pg_database
From backend> C
From backend> "SELECT"
To backend> Q
From backend> I
From backend>
To backend> QFETCH all in cursor_1
From backend> P
From backend> "cursor_1"
From backend> T
From backend (#2)> 3
From backend> "datname"
From backend (#4)> 19
From backend (#2)> 32
From backend> "datdba"
From backend (#4)> 23
From backend (#2)> 4
From backend> "datpath"
From backend (#4)> 25
From backend (#2)> 65535
From backend> D
From backend (1)> à
From backend (#4)> 13
From backend (9)> template1
From backend (#4)> 7
From backend (3)> 500
From backend (#4)> 13
From backend (9)> template1
From backend (1)> à
From backend (#4)> 7
From backend (3)> bd1
From backend (#4)> 7
From backend (3)> 500
From backend (#4)> 7
From backend (3)> bd1
From backend> "FETCH"
To backend> QCLOSE cursor_1
From backend> C
From backend> "CLOSE"
To backend> Q
From backend> I
From backend>
To backend> QEND
From backend> C
From backend> "END"
To backend> Q
From backend> I
From backend>
```

da se obtiene como resultado de la funcion **PGexec** y es utilizada por las demás funciones de ejecución.

La práctica que se va a realizar efectuará una lectura de la tabla **tabla1** de la base **bd1** que se ha definido y cargado previamente con los procedimientos SQL **creatabla** e **insert** que se suministran en el CD-ROM. También es posible obtener los datos de la tabla ejecutando el procedimiento **select.sql**.

Para parar el postmaster se puede usar el comando kill, evitando dar la opción –9 o –KILL

El programa ejecutará la misma sentencia **select** que viene en el procedimiento incluido.

El primer paso para programar en PostGres consiste en añadir el fichero de cabecera **libpq-fe.h** tal y como se ha indicado anteriormente. En este fichero de cabecera están definidas las estructuras de tipo **PGconn** y **PGres**, de las que es necesario definir sendas variables:

```
PGconn *conn;
PGres  *res;
```

Por simplicidad se va dejar que la conexión se realice con los valores por defecto por lo que las variables **pgport**, **pghost**, etc. se ponen con el calor NULL, a excepción del nombre de la base datos.

La conexión se realiza con la función **PQsetdb** de la forma:

```
conn = PQsetdb(pgpst,pgport, pgoptions, pgtty,
     mpmbreDB)
```

El resultado de esta función, si es correcta, es una estructura de tipo **PGconn**. El resultado de la conexión se verifica con la función **PGstatus**. Si la conexión se ha realizado satisfactoria-

mente, se establece una conexión por *socket* entre el programa y el servidor, al igual que cuando se accede por medio de **psql**. El programa tiene una pausa para que se pueda desde otra sesión **telnet** ejecutar el comando:

```
netstat –an | grep 5432
```

Y observar la conexión establecida entre los dos procesos. Si el servidor y el cliente están en el mismo *host*, aparecerán dos conexiones establecidas con las direcciones local y remota permutadas.

Las siguientes instrucciones permiten ejecutar el siguiente grupo de sentencias SQL: *BEGIN, DECLARE CURSOR, FETCH, CLOSE, END*.

Todas ellas se ejecutan con el API de la misma forma:

```
res = PQexec(conn,"sentencia");
if (PqresultStatus(res) i=
      PGRES_COMMAND_OK)
{
      < tratamiento error>
}
```

Como se puede observar la función **PQexec** hace referencia a la estructura **PGconn**, lo que permite desde un mismo programa gestionar varias conexiones con distintas bases de datos e incluso con distintos servidores, simplemente con tener varias estructuras **PGconn** activadas.

TRABAJANDO CON SQL

Aunque no es el objetivo de este artículo profundizar en el lenguaje SQL se va a explicar las sentencias que se manejan en el programa.

La sentencia BEGIN, inicia una transacción SQL, que es una unidad de programación indivisible, es decir una transacción se ejecuta totalmente o no se ejecuta. El diseño de las transacciones es crítico para gestionar los accesos concurrentes a las bases de datos así como la recuperación de errores.

La siguiente sentencia declara un cursor, un cursor es un objeto de las bases de datos relacionales que permite moverse lógicamente por el resultado de una *query*. En la sentencia el cursor se declara relacionado con una **select:**

```
DECLARE cursor_1 CURSOR FOR select * from
      tabla1
```

Para obtener el resultado de la sentencia **select** asociada al cursor hay ejecutar una instrucción **FETCH**. Si el resultado de esta instrucción es **PG_TUPLES_OK**, se puede utilizar las funciones **PQntuples, PQnfields, PQgetvalue**, etc. para obtener información acerca de los datos obtenidos.

La creación de directorios para las bases de datos se realiza por medio del comando initdb

El programa construye la cabecera de la salida con los nombres de los campos, obteniendo primero el número de campos que existen en el cursor, en este caso coincide con el número de campos existente en la tabla. A continuación va obteniendo los nombres de cada uno de los campos con la función **Pqfname**. Esta función tiene dos parámetros el primero es un puntero a la estructura **PGres**, y el segundo es un índice a la tabla de campos del cursor:

La creación de tablas que contienen atributos que son grandes objetos se realiza por medio del tipo de datos oid

El prototipo de la función **PQfname** es el siguiente:

```
char * PQfname( *PGres, int num_atrib).
```

Para obtener los valores de cada instancia de la tabla se utiliza la función **PQgetvualue** cuyo prototipo es:

```
char * PQgetvalue( *PGres, int num_instancia, int
      num_atrib)
```

En esta función la variable de tipo **char** no hay que entenderla como una variable de caracteres, si no como un **string** de octetos terminados por el carácter nulo.

Pueden tener o no tener representación alfanumérica, quedando a la responsabilidad del programador realizar el tratamiento adecuado según el tipo de variable que corresponda.

```
[juanma@localhost juanma]$ prog
Conexion realizada con la base de datos: bd1
Equipo: localhost
Puerto de conexion: 5432
Terminal tty:
Opciones de conexion:

Pausa establecida para poder observar las conexiones
Pulse intro para continuar...

campo11     campo12     campo13
----------- ----------- ----------
1           texto111    01-01-1998
1           texto111    01-01-1998
1           texto112    01-01-1998
1           texto113    01-01-1998
1           texto114    01-01-1998
1           texto115    01-01-1998
[juanma@localhost juanma]$
```

Figura 8. Resultado de la ejecución del programa prog. Este programa lista el contenido de la tabla1.

Una vez que se ha realizado el proceso, se cierra el cursor con **CLOSE**, la transacción con **END** y la conexión con la función **PQfinish.**

PostGres dispone de un API de programación implementado en la biblioteca libpq.a. que se instala en el paquete de desarrollo

La estructura **PGres** es reutilizada en cada **Pqexec**. Para conseguir que una misma estructura pueda ser utilizada por distintas **PQexec** hay que limpiar dicha estructura con la función **PQClear**.

LOS CATÁLOGOS DE SISTEMA

Como ya se explicó anteriormente, en la estructura de tablas contenidas en el directorio de base de datos de PostGres existen una serie de tablas cuyo nombre comienza por **pg**. Estas tablas contienen información acerca del sistema. En el cuadro 8 se pueden ver algunas de las tablas que forman el catálogo del mismo. Al ser tablas el propio catálogo del sistema puede ser accedido mediante SQL. El programa **prog1.c**, que también se incluye en el CD-ROM, es una versión modificada del programa anterior, con dos diferencias, el nombre de la base de datos pasa a ser **template1**. Aunque en este caso no se va a trabajar con ninguna base de datos en concreto es necesario especificar un nombre de base de datos para establecer la conexión con el servidor. No es necesario que el nombre de la base de datos sea **template1** pero esta base de datos normalmente existe en cualquier instalación de PostGres, así queda asegurado el resultado correcto de la conexión. La otra diferencia es el nombre de la tabla en la sentencia **select** de la declaración del cursor que cambia a **pg_database**, que es la tabla que contiene la descripción de todas la bases de datos gestionadas por ese *postmaster*.

```
[juanma@localhost juanma]$ prog1
datname         datdba          datpath
--------------- --------------- ---------------
template1       500             template1
bd1             500             bd1
[juanma@localhost juanma]$
```

Figura 9. Resultado de la ejecución del programa prog1.

DEPURACIÓN Y RASTREO DE PROBLEMAS

PostGres suministra dos funciones que permite la depuración de programas. Estas funciones son **PQtrace** y **PQuntrace**. La primera inicia un *trace* del flujo de datos entre el programa que se está depurando (*frontend*) y en servidor de datos (*backend*). El prototipo de esta función es:

```
void PQtrace (PGconn *conn, FILE *fichero)
```

La salida del *trace* se almacena en el fichero cuyo nombre se indica en el segundo parámetro. El primer parámetro es un puntero a una estructura **PGconn**, previamente inicializada. Es decir, sólo se puede arrancar una sesión de *trace* una vez que se ha realizado la conexión con la base de datos mediante la función **PQsetdb**.

Para cerrar la sesión de *trace* se usa la función **PQuntrace** cuyo prototipo es:

```
void PQuntrace(PGconn *conn)
```

La sesión de *trace* hace también referencia a una estructura **PGconn** inicializada. Si se cierra la conexión con la base de datos con la función **PQfinish** la sesión de *trace* finaliza también y cualquier intento de cerrar la sesión de *trace* será ignorado por el sistema.

El programa **prog2** del CD-ROM es igual al programa **prog1** que lee la tabla **pg_database** y se ha añadido un arranque y parada de una sesión de *trace*. El fichero de salida de *trace* ha de ser abierto explícitamente antes de iniciar el proceso, pues la función **PQtrace** no realiza esta tarea. De igual forma, una vez finalizado el *trace* debe de ser explícitamente cerrado.

Es necesario suministrar los valores: PGHOST, PGOPTIONS, PGPORT, PGTTY, PGDATABSE

En el cuadro 9 se puede observar el resultado obtenido al ejecutar el programa **prog2**. Las líneas que tiene como etiqueta **<to backend>** representan el flujo generado desde el programa al servidor y el etiquetado como **<From backend>** el de sentido contrario. Se puede observar como aparecen en el fichero tanto las sentencias SQL que manda el *frontend* como las respuestas de datos que envía el *backend* más una serie de comandos de control del protocolo que existen entre ambos extremos de la conexión. Con el programa y el fichero listado por un lado y la salida de la ejecución sobre la pantalla por otro, es posible seguir la lógica completa de la sesión.

Las funciones de trace son extremadamente gravosas para el rendimiento del sistema por lo que sólo de deben utilizar para determinar la causa de algún problema

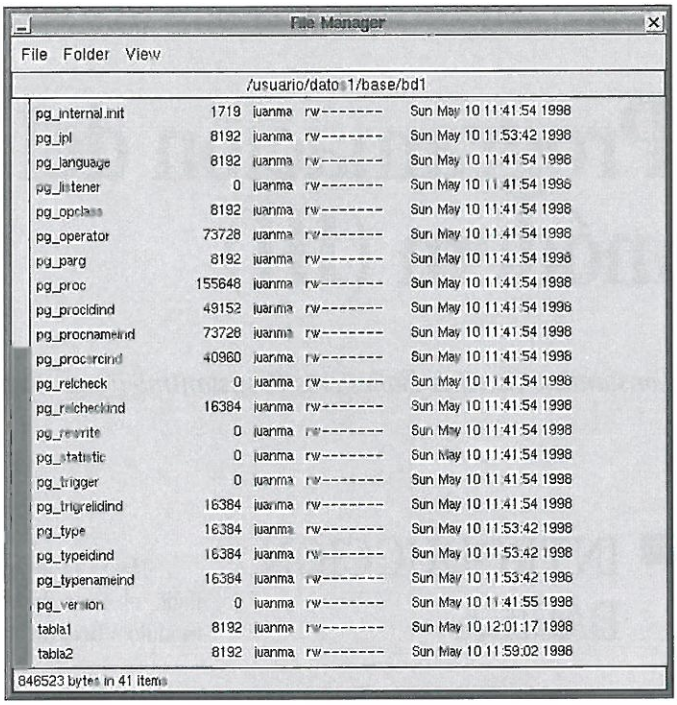

File Manager
File Folder View
/usuario/datos1/base/bd1
pg_internal.init 1719 juanma rw------- Sun May 10 11:41:54 1998
pg_ipl 8192 juanma rw------- Sun May 10 11:53:42 1998
pg_language 8192 juanma rw------- Sun May 10 11:41:54 1998
pg_listener 0 juanma rw------- Sun May 10 11:41:54 1998
pg_opclass 8192 juanma rw------- Sun May 10 11:41:54 1998
pg_operator 73728 juanma rw------- Sun May 10 11:41:54 1998
pg_parg 8192 juanma rw------- Sun May 10 11:41:54 1998
pg_proc 155648 juanma rw------- Sun May 10 11:41:54 1998
pg_procidind 49152 juanma rw------- Sun May 10 11:41:54 1998
pg_procnameind 73728 juanma rw------- Sun May 10 11:41:54 1998
pg_procrcind 40960 juanma rw------- Sun May 10 11:41:54 1998
pg_relcheck 0 juanma rw------- Sun May 10 11:41:54 1998
pg_relcheckind 16384 juanma rw------- Sun May 10 11:41:54 1998
pg_rewrite 0 juanma rw------- Sun May 10 11:41:54 1998
pg_statistic 0 juanma rw------- Sun May 10 11:41:54 1998
pg_trigger 0 juanma rw------- Sun May 10 11:41:54 1998
pg_trigrelidind 16384 juanma rw------- Sun May 10 11:41:54 1998
pg_type 16384 juanma rw------- Sun May 10 11:53:42 1998
pg_typeidind 16384 juanma rw------- Sun May 10 11:53:42 1998
pg_typenameind 16384 juanma rw------- Sun May 10 11:53:42 1998
pg_version 0 juanma rw------- Sun May 10 11:41:55 1998
tabla1 8192 juanma rw------- Sun May 10 12:01:17 1998
tabla2 8192 juanma rw------- Sun May 10 11:59:02 1998
846523 bytes in 41 items

Figura 10. Listando el contenido del directorio -D del postmaster se pueden ver las distintas tablas del sistema.

que tenga el programa y desactivar el *trace* una vez solucionado. Para evitar tener que compilar el programa para activar el *trace,* una buena práctica de programación es dejar la activación en manos de un dato externo, bien sea un parámetro de arranque del programa o una variable de entorno.

Se puede dejar la activación del trace en función de una variable:

```
var = getenv("PGTRACE")
if ((i = strcpy(PGRACE,"SI)) == 0)
then {
tracefile = fopen("./trace.file","w");
PQtrace(conn,tracefile);
    }
```

Y realizar la llamada al programa de la siguiente forma:

```
PGTRACE=SI
export PGTRACE
prog
```

MIGRACIÓN

En el momento de escribir este artículo ya está disponible la versión 6.3 de PostGres. Debido a sus particulares características, PostGres no mantiene entre versiones el mismo formato para sus datos de una versión a otra, por lo que es necesario realizar una migración de los datos y extensiones (tipos definidos por el usuario, funciones, etc).

En general se deben realizar los siguientes pasos:

- *Extensiones*: Se crean de nuevo.
- *Datos*: se vuelcan a ficheros ASCII con el comando **COPY**, se crean las nuevas tablas y se vuelven a cargar las tablas.
- *Reglas y vistas*: se crean de nuevo.

En los tres casos se puede ver la importancia que tiene ir guardando todas las sentencias SQL que se han utilizado para la creación de todos los objetos existentes en las bases de datos.

Estos pasos se deben tomar como norma general, ya que en cada cambio de *release* pueden existir peculiaridades, por lo que se aconseja leer todas las notas relacionadas con la nueva versión 6.3 anteriormente citada en el artículo, y las guías de migración que las acompañan.

CONCLUSIÓN

No quisiera finalizar el artículo sin expresar una valoración personal acerca de este sistema de bases de datos que tiene una funcionalidad al menos igual a otras bases de datos profesionales existentes en el mercado.Como casi todos los productos Linux carece de una interfaz de usuario cómoda, aunque existe una herramientas de gestión de PostGres en modo gráfico con X-Windows llamada **pgaccess** que ofrece una mayor potencia a la hora de la gestión de este producto.

Por otra parte la instalación es muy fácil, y si el paquete viene el formato rpm se realiza de forma automática y sin requerir ningún tipo de acción especial. El *postmaster* queda arrancado y simplemente con ejecutar el comando **pgsql template1**, se puede empezar a trabajar con la base de datos. Aunque como se ha dicho la interfaz no es muy amigable, el diseño del producto permite realizar las tareas de administración y explotación con bastante comodidad. Entre estas tareas de administración que se pueden realizar de forma casi directa están:

- Distribución de las bases de datos entre distintos servidores.
- Posibilidad de tener varios servidores en un mismo equipo.
- No es necesario la existencia de una máquina dedicada para pruebas de nuevas versiones.

BIBLIOGRAFÍA Y WEBS

- Manual de Usuario de PostGres (incluido el formato *postscript* en el paquete del programa.
- PostGresSQL Programmers's Guide.
- Dirección de la página Web: www.postgresql.org

PROGRAMACIÓN HARDWARE

Programación del módem (III)

Constantino Sánchez Ballesteros (Constantino@redestb.es)

A largo de este artículo mantendremos la evolución de la programación del módem que comenzamos hace dos entregas.

En esta ocasión conoceremos la posibilidad de una conexión entre la línea telefónica y el módem.

INTRODUCCIÓN BÁSICA

Vamos a continuar el desarrollo del programa que comenzamos en el fascículo anterior. Se trataba de un programa definido como "terminal". Este tipo de aplicaciones es muy utilizado por BBS's para la transferencia de ficheros, así como el envío y recepción de mensajes.

El programa principal, constaba de los siguientes módulos y formularios en el proyecto:

Módulos

- **Vbterm.glo:** declara ciertas variables y llama a una DLL del API Win32.

Formularios

- **Cansend.frm:** es un cuadro de diálogo que permite al usuario cancelar una operación "Transmitir archivo de texto". Se trata de un formulario no modal que actúa como modal permitiendo continuar a otros procesos.
- **Frmprops.frm:** es el formulario encargado de establecer las configuraciones por defecto.
- **Vbterm.frm:** es el formulario principal que muestra la interface del programa.

Hasta el momento, hemos desarrollado el formulario **Cansend.frm** y el módulo **Vbterm.glo**

FORMULARIO FRMPROPERTIES (FRMPROPS.FRM)

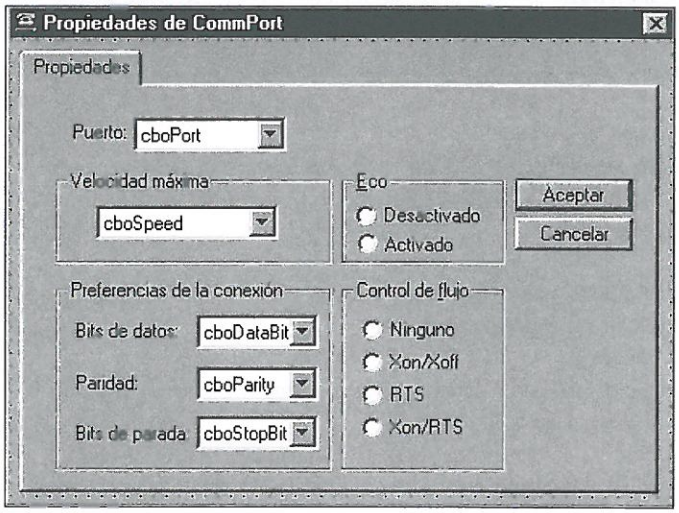
Este nuevo formulario, establecerá las propiedades de conexión con la línea telefónica y el módem. El primer paso es establecer un par de variables privadas que nos serán útiles dentro de este módulo de formulario para asignar el flujo de datos y valores temporales de Eco.

```
Private iFlow As Integer, iTempEcho As Boolean
```

El siguiente procedimiento nos servirá para ajustar las configuraciones según la elección del usuario, cuando determine sus preferencias en las diferentes selecciones del formulario (frmproperties) representado a continuación:

```
Sub LoadPropertySettings()
Dim i As Integer, Settings As String, Offset As
    Integer

    ' Carga las configuraciones de puerto
For i = 1 To 16
  cboPort.AddItem "Com" & Trim$(Str$(i))
Next i
```

```
' Carga las configuraciones de velocidad
cboSpeed.AddItem "110"
cboSpeed.AddItem "300"
cboSpeed.AddItem "600"
cboSpeed.AddItem "1200"
cboSpeed.AddItem "2400"
cboSpeed.AddItem "4800"
cboSpeed.AddItem "9600"
cboSpeed.AddItem "14400"
cboSpeed.AddItem "19200"
cboSpeed.AddItem "28800"
cboSpeed.AddItem "38400"
cboSpeed.AddItem "56000"
cboSpeed.AddItem "57600"
cboSpeed.AddItem "115200"
cboSpeed.AddItem "128000"
cboSpeed.AddItem "256000"

' Carga las configuraciones de bits de datos
cboDataBits.AddItem "4"
cboDataBits.AddItem "5"
cboDataBits.AddItem "6"
cboDataBits.AddItem "7"
cboDataBits.AddItem "8"

' Carga las configuraciones de paridad
cboParity.AddItem "Even"
cboParity.AddItem "Odd"
cboParity.AddItem "None"
cboParity.AddItem "Mark"
cboParity.AddItem "Space"

' Carga las configuraciones de bits de parada
cboStopBits.AddItem "1"
cboStopBits.AddItem "1.5"
cboStopBits.AddItem "2"

' Establece la configuración predeterminada
Settings = frmTerminal.MSComm1.Settings

' En todos los casos, el componente más a la derecha de Settings

' será un solo carácter, excepto cuando haya 1,5 bits de parada.
If InStr(Settings, ".") > 0 Then
  Offset = 2
Else
  Offset = 0
End If
```



Cuadro de propiedades de Commport.

```
cboSpeed.Text = Left$(Settings, Len(Settings) - 6 - Offset)

Select Case Mid$(Settings, Len(Settings) - 4 - Offset, 1)
Case "e"
  cboParity.ListIndex = 0
Case "m"
  cboParity.ListIndex = 1
Case "n"
  cboParity.ListIndex = 2
Case "o"
  cboParity.ListIndex = 3
Case "s"
  cboParity.ListIndex = 4
End Select

cboDataBits.Text = Mid$(Settings, Len(Settings) - 2 - Offset, 1)
cboStopBits.Text = Right$(Settings, 1 + Offset)

cboPort.ListIndex = frmTerminal.MSComm1.CommPort - 1

optFlow(frmTerminal.MSComm1.Handshaking).Value = True

If Echo Then
  optEcho(1).Value = True
Else
  optEcho(0).Value = True
End If

End Sub
```

Este procedimiento servirá para descargar de memoria el formulario cuando se pulse el botón "Cancelar". Para ello, utilizaremos la función estándar de Visual Basic *Unload*.

```
Private Sub cmdCancel_Click()
Unload Me
End Sub
```

Al pulsar sobre el botón "Aceptar" se generará este procedimiento, el cual se encarga de activar los datos asignados en las propiedades visuales del formulario. La variable *Oldport* tiene establecida la configuración actual del puerto hasta que no se indique lo contrario; si es el caso, el nuevo valor se establecerá en *Newport*.

```
Private Sub cmdOK_Click()
Dim OldPort As Integer, ReOpen As Boolean

On Error Resume Next

Echo = iTempEcho
OldPort = frmTerminal.MSComm1.CommPort
NewPort = cboPort.ListIndex + 1

If NewPort <> OldPort Then  ' Si cambia el número de puerto, cierra el antiguo.
  If frmTerminal.MSComm1.PortOpen Then
    frmTerminal.MSComm1.PortOpen = False
    ReOpen = True
  End If

  frmTerminal.MSComm1.CommPort = NewPort
    ' Establece el nuevo número de puerto.

  'Abrimos el puerto
  If Err = 0 Then
```

Tabla 1. Sintaxis de SaveSetting .

Sintaxis

SaveSetting appname, section, key, setting

La sintaxis de la instrucción SaveSetting tiene los siguientes argumentos con nombre:

Parte	Descripción
appname	Obligatorio. Expresión de cadena que contiene el nombre de la aplicación o proyecto al que se aplica el valor.
section	Obligatorio. Expresión de tipo cadena que contiene el nombre de la sección donde se va a guardar la clave.
key	Obligatorio. Expresión de tipo cadena que contiene el nombre de la clave que se va a guardar.
setting	Obligatorio. Expresión que contiene el valor que se va a establecer en *key*.

Nota: *Puede producirse un error si por alguna razón no se puede guardar el valor de la clave. Los valores asignados también pueden verse a través del programa Regedit.exe de Windows.*

```
    If ReOpen Then
      frmTerminal.MSComm1.PortOpen = True
      frmTerminal.mnuOpen.Checked =
     frmTerminal.MSComm1.PortOpen
  frmTerminal.mnuSendText.Enabled =
     frmTerminal.MSComm1.PortOpen
frmTerminal.tbrToolBar.Buttons("TransmitTextFile
     ").Enabled =
     frmTerminal.MSComm1.PortOpen
    End If
  End If
```

'Si se establece un error al abrir el puerto, se asignará el valor del antiguo (*Oldport*)

```
  If Err Then
    MsgBox Error$, 48
    frmTerminal.MSComm1.CommPort =
     OldPort
    Exit Sub
  End If
End If
```

'Asignamos los valores de velocidad, paridad, bits de datos y de parada

```
frmTerminal.MSComm1.Settings =
      Trim$(cboSpeed.Text) & "," &
      Left$(cboParity.Text, 1) _
  & "," & Trim$(cboDataBits.Text) & "," &
      Trim$(cboStopBits.Text)
If Err Then
  MsgBox Error$, 48
  Exit Sub
End If

frmTerminal.MSComm1.Handshaking = iFlow
If Err Then
  MsgBox Error$, 48
  Exit Sub
End If
```

En las siguientes líneas del procedimiento se utiliza la instrucción estándar de Visual Basic "*SaveSetting*". Esta instrucción Guarda o crea una entrada para una aplicación en el registro de configuración de Windows. La siguiente tabla muestra sus parámetros para que tengáis una referencia rápida:

```
SaveSetting App.Title, "Properties", "Settings",
      frmTerminal.MSComm1.Settings
SaveSetting App.Title, "Properties", "CommPort",
      frmTerminal.MSComm1.CommPort
SaveSetting App.Title, "Properties",
      "Handshaking",
      frmTerminal.MSComm1.Handshaking
SaveSetting App.Title, "Properties", "Echo", Echo
```

'Descargamos el formulario

```
Unload Me

End Sub
```

Este procedimiento se inicia cuando se carga el formulario en pantalla. Definiremos propiedades del mismo así como las propiedades de comunicación establecidas o por defecto.

```
Private Sub Form_Load()
```

' Establece el tamaño del formulario

```
Me.Left = (Screen.Width - Me.Width) / 2
Me.Top = (Screen.Height - Me.Height) / 2
```

' Ajusta el tamaño del marco al control tabstrip

```
fraSettings.Move tabSettings.ClientLeft,
      tabSettings.ClientTop
```

' Se asegura de que el marco sea el control situado encima

```
fraSettings.ZOrder
```

' Carga la configuración actual de las propiedades

```
LoadPropertySettings

End Sub
```

Activamos o desactivamos el Eco.

```
Private Sub optEcho_Click(Index As Integer)
If Index = 1 Then
  iTempEcho = True
Else
  iTempEcho = False
End If
End Sub
```

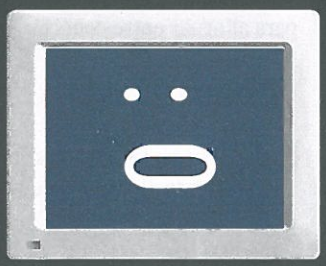
Seleccionamos el control de flujo a utilizar.

```
Private Sub optFlow_Click(Index As Integer)
iFlow = Index
End Sub
```

FORMULARIO FRMTERMINAL (VBTERM.FRM)

Para finalizar nuestro proyecto, tan sólo nos queda comentar el formulario principal que visualiza la interfaz al usuario para la utilización del programa. *Este formula-*

¿INTERNET ASEQUIBLE?



SÍ, ASEQUIBLE

NO ENCONTRARÁS UNA OFERTA MEJOR

¿Cómo?

Con un completo curso multimedia que te explicará desde en qué consiste el correo electrónico hasta qué es el FTP. Y con conexión gratuita a Internet hasta el 31 de diciembre de 1.998.

¿Cuánto?

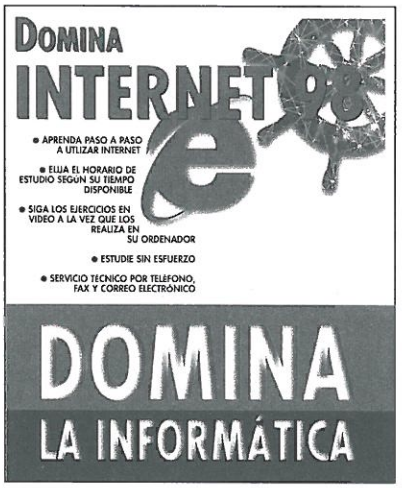
Por sólo 3.995 ptas. Iva incluido.

¿Dónde?

Podrás encontrarlo en librerías, quioscos, tiendas de informática, Vips, Crisol, Fnac, El Corte Inglés, Casa del libro o en el teléfono 91-6614211.

¿Cuándo?

Ya a la venta.


DOMINA
INTERNET
• APRENDA PASO A PASO A UTLIZAR INTERNET
• ELIJA EL HORARIO DE ESTUDIO SEGÚN SU TIEMPO DISPONIBLE
• SIGA LOS EJERCICIOS EN VIDEO A LA VEZ QUE LOS REALIZA EN SU ORDENADOR
• ESTUDIE SIN ESFUERZO
• SERVICIO TÉCNICO POR TELÉFONO, FAX Y CORREO ELECTRÓNICO
DOMINA
LA INFORMÁTICA

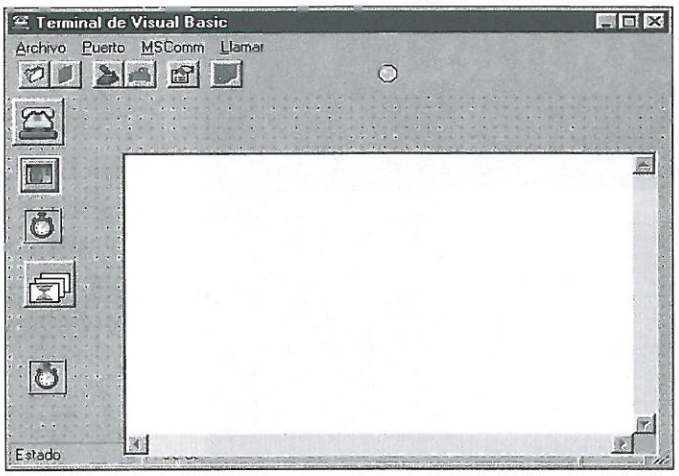
ABETO editorial

C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Tel.: (91) 661 42 11*
Fax: (91) 661 43 86
http://www.abetoed.es

También versión sin conexión por 2.995 ptas.

rio será comentado en este fascículo y el siguiente debido a su longitud...

```
Option Explicit

Dim Ret As Integer      ' Entero auxiliar.
Dim Temp As String      ' Cadena auxiliar.
Dim hLogFile As Integer ' Controlador de
    archivo de registro abierto.
Dim StartTime As Date   ' Almacena la hora
    de inicio del cronómetro del puerto

Private Sub Form_Load()
  Dim CommPort As String, Handshaking As
    String, Settings As String

  On Error Resume Next

    ' Establece el color predeterminado
del terminal
  txtTerm.SelLength = Len(txtTerm)
  txtTerm.SelText = ""
  txtTerm.ForeColor = vbBlue

    ' Establece el título
  App.Title = "Terminal de Visual Basic"

    ' Establece la luz indicadora de estado
  imgNotConnected.ZOrder

    ' Centra el formulario
  frmTerminal.Move (Screen.Width - Width) / 2,
    (Screen.Height - Height) / 2

    ' Carga la configuración del registro

  Settings = GetSetting(App.Title, "Properties",
    "Settings", "") '
    frmTerminal.MSComm1.Settings]\
  If Settings <> "" Then
    MSComm1.Settings = Settings
    If Err Then
      MsgBox Error$, 48
      Exit Sub
    End If
  End If

  CommPort = GetSetting(App.Title,
    "Properties", "CommPort", "") '
    frmTerminal.MSComm1.CommPort
  If CommPort <> "" Then
    MSComm1.CommPort = CommPort

  Handshaking = GetSetting(App.Title,
    "Properties", "Handshaking", "")
```



Cuadro de terminal de Visual Basic.

```
    'frmTerminal.MSComm1.Handshaking
  If Handshaking <> "" Then
    MSComm1.Handshaking = Handshaking
    If Err Then
      MsgBox Error$, 48
      Exit Sub
    End If
  End If

  Echo = GetSetting(App.Title, "Properties",
    "Echo", "") ' Echo
  On Error GoTo 0

End Sub

Private Sub Form_Resize()
    ' Cambia el tamaño del control Term
(ventana)
  txtTerm.Move 0, tbrToolBar.Height,
    frmTerminal.ScaleWidth,
    frmTerminal.ScaleHeight - sbrStatus.Height
    - tbrToolBar.Height

    ' Sitúa la luz indicadora de estado
  Frame1.Left = ScaleWidth - Frame1.Width * 1.5
End Sub

Private Sub Form_Unload(Cancel As Integer)
  Dim Counter As Long

  If MSComm1.PortOpen Then
    ' Se establece una espera de 10
segundos para transmitir los datos.
    Counter = Timer + 10
    Do While MSComm1.OutBufferCount
      Ret = DoEvents()
      If Timer > Counter Then
        Select Case MsgBox ("Imposible el
        envío de datos al receptor", 34)
        ' Cancelar.
        Case 3
          Cancel = True
          Exit Sub
        ' Reintentar.
        Case 4
          Counter = Timer + 10
        ' Ignorar.
        Case 5
          Exit Do
        End Select
      End If
    Loop

    MSComm1.PortOpen = 0
  End If

    ' Si el archivo de registro está abier-
to, vuelca y lo cierra.
  If hLogFile Then mnuCloseLog_Click
  End
End Sub

Private Sub imgConnected_Click()
    ' Llama a la rutina mnuOpen_Click
para alternar entre conectar y desconectar.
  Call mnuOpen_Click
End Sub

Private Sub imgNotConnected_Click()
    ' Llama a la rutina mnuOpen_Click
para alternar entre conectar y desconectar.
  Call mnuOpen_Click
End Sub

Private Sub mnuCloseLog_Click()
    ' Cierra el archivo de registro.
  Close hLogFile
  hLogFile = 0
```

```
    mnuOpenLog.Enabled = True
    tbrToolBar.Buttons("OpenLogFile").Enabled =
        True
    mnuCloseLog.Enabled = False
    tbrToolBar.Buttons("CloseLogFile").Enabled =
        False
    frmTerminal.Caption = "Terminal de Visual
        Basic"
End Sub

Private Sub mnuDial_Click()
    On Local Error Resume Next
    Static Num As String

    Num = "055" ' Aquí asignamos el núme-
        ro a marcar por defecto.

    ' Obtiene un número del usuario.
    Num = InputBox$("Escriba el número de telé-
        fono:", "Marcar número", Num)
    If Num = "" Then Exit Sub

    ' Abre el puerto si no lo está ya.
    If Not MSComm1.PortOpen Then
        mnuOpen_Click
        If Err Then Exit Sub
    End If

    ' Habilita el botón de colgar y el ele-
mento de menú correspondiente.
    mnuHangUp.Enabled = True
    tbrToolBar.Buttons("HangUpPhone").Enabled =
        True

    ' Marca el número.
    MSComm1.Output = "ATDT" & Num & vbCrLf

    ' Inicia el cronómetro del puerto.
    StartTiming
End Sub

Private Sub mnuDTREnable_Click()
    ' Alterna la propiedad DTREnable
    MSComm1.DTREnable = Not
        MSComm1.DTREnable
    mnuDTREnable.Checked =
        MSComm1.DTREnable
End Sub

Private Sub mnuFileExit_Click()
    ' Utiliza Form_Unload, ya que tiene
código para comprobar si hay datos
    ' no enviados y si hay un archivo de
registro abierto.
    Form_Unload Ret
    Sub

    ' Alterna la propiedad DTREnable
para colgar la línea.
Private Sub mnuHangup_Click()
    On Error Resume Next

    MSComm1.Output = "ATH"      ' Envía la
        cadena de colgar
    Ret = MSComm1.DTREnable     ' Guarda el
        valor actual.
    MSComm1.DTREnable = True    ' Activa
        DTR.
    MSComm1.DTREnable = False   ' Desactiva
        DTR.
    MSComm1.DTREnable = Ret     ' Restablece
        el valor anterior.

    mnuHangUp.Enabled = False
    tbrToolBar.Buttons("HangUpPhone").Enabled =
        False

    ' Si el puerto continúa abierto, lo
cierra
    If MSComm1.PortOpen Then
        MSComm1.PortOpen = False

    ' Notifica el error al usuario
    If Err Then MsgBox Error$, 48

    mnuSendText.Enabled = False
    tbrToolBar.Buttons("TransmitTextFile").Enabled
        = False
    mnuHangUp.Enabled = False
    tbrToolBar.Buttons("HangUpPhone").Enabled =
        False
    mnuDial.Enabled = True

        tbrToolBar.Buttons("DialPhoneNumber").En
        abled = True
    sbrStatus.Panels("Settings").Text = "Valores:"

    ' Apaga la luz indicadora y quita la
marca al menú
    mnuOpen.Checked = False
    imgNotConnected.ZOrder

    ' Detiene el cronómetro del puerto
    StopTiming
    sbrStatus.Panels("Status").Text = "Estado:"
    On Error GoTo 0
End Sub
```

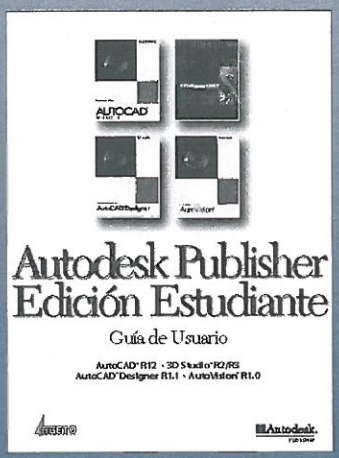
AUTODESK PUBLISHER EDICIÓN ESTUDIANTE

¿QUÉ PROGRAMAS INCLUYE?

- AutoCAD R12
- 3D Studio R2/R3
- AutoCAD Designer R1.1
- Autovision R1.0.

Y además:

Más de 700 páginas de lecciones de Autodesk con guías de referencia y documentación software on-line



Formato 21x29,7

TODO LO QUE NECESITA PARA EL DISEÑO Y VISUALIZACIÓN PROFESIONAL EN 2D Y 3D LO HALLARÁ EN ESTE PAQUETE DE PROGRAMAS.

EDICIÓN ESPECIAL PARA ESTUDIANTES
19.500 PTAS. (I.V.A. INCLUIDO)

DE VENTA EN LIBRERÍAS Y DISTRIBUIDORES AUTODESK

PROGRAMACIÓN DE SOCKETS

Creación de un servidor Web con Delphi (II)

Enrique de la Lastra (elastra@redestb.es)

¿Cómo envía un servidor web las páginas HTML que solicita un navegador?. En este artículo lo descubriremos y mediante Delphi crearemos un servidor de este tipo. Además funcionará como un servidor del protocolo echo.
Esta es la pregunta que intentaremos resolver en las siguientes páginas.

INTRODUCCIÓN

En el artículo del mes pasado nos adentramos en el mecanismo de funcionamiento del protocolo *HTTP* (*HyperText Transfer Protocol*) y aprendimos los conceptos básicos necesarios para introducirnos en la programación de un servidor web que proporcione el servicio básico de entrega de páginas HTML, junto con los objetos en ellas referenciados.

Para comenzar con la programación del servidor, nos familiarizamos con el componente Delphi *TServerSocket* cuya función es la de convertir nuestra aplicación en un servidor *TCP/IP*. Sobre este componente ya creamos una aplicación servidora del protocolo *echo*, así como un cliente del mismo protocolo que permitiera probar el servidor.

El servicio *echo* que creamos se puede añadir fácilmente a nuestro Servidor Web sin más que añadir a la aplicación entonces creada un *socket* que atienda las peticiones en el puerto *http* (número **80**). De igual manera, podemos integrar en la aplicación servidora que vayamos a crear tantos servicios de Internet como queramos: *FTP* (*File Transfer Protocol*, Protocolo de Transferencia de Ficheros), *SMTP* (*Simple Mail Transfer Protocol*, Protocolo Simple de Transferencia de Correo), etc.

A partir de aquí veremos cómo crear nuestro propio servidor web e integraremos en él, por el momento, el servicio de *echo*. No entraremos en la explicación de los fundamentos del *HTTP* (véase el anterior artículo), sino más bien en la explicación de las cabeceras y los métodos definidos en la especificación que vamos a utilizar. El servidor que crearemos proporcionará las funciones básicas de un servidor seb: entregar páginas HTML a los clientes.

En cuanto a las limitaciones, el servidor no tratará secuencias *CGI* y estará limitado a la versión 1.0 del protocolo *HTTP*. Por tanto, ignorará las peticiones simples (HTTP/0.9) y no va a atender a los nuevos métodos de petición definidos en la HTTP/1.1 (aunque sí hará caso de los métodos definidos en la especificación 1.1 compatibles con la versión HTTP/1.0). Como limitación adicional, será incapaz, en esta primera versión, de aceptar varias peticiones de distintos clientes al tiempo, de forma que cada petición de un nuevo cliente debe esperar a que la anterior se haya cursado completamente.

Para que el lector pueda profundizar en la especificación del protocolo HTTP, se incluyen en el CD-ROM de la revista las *RFC's* (*Requests For Comment*, Solicitudes Para Comentar) del protocolo *HTTP* en sus versiones 1.0 y 1.1 (**rfc1945** y **rfc2068**, respectivamente). Se incluyen otras *RFC's* relacionadas con el protocolo *HTTP* que también pueden resultar interesantes.

PETICIONES HTTP

Entramos de lleno estudiando el formato de las peticiones *HTTP*, es decir del texto que vamos a recibir de un navegador (como, por ejemplo, Netscape o Internet Explorer).

De acuerdo con la especificación 1.0 del protocolo *HTTP* (**rfc1945**), la notación de un método de petición HTTP es la siguiente:

```
Método URI VersionDelProtocolo #13#10
[*<Cabecera>]
[#13#10 <datos>]
```

En esta notación, las partes entre corchetes son opcionales; el asterisco indica que el parámetro a continuación puede consistir de uno o más ítems; los símbolos < > representan un campo compuesto de varias partes; y aunque aquí no aparece, el símbolo "|" es equivalente a una "O" (es decir "A | B", quiere decir que puede ser: A ó B ó ambas).

Veamos uno a uno los componentes de la petición:

- **Método** = Método de petición. En la especificación *HTPP/1.0* se definen tres métodos: *GET*, HEAD y *POST*. Se definen también cuatro métodos adicionales que no deben ser necesariamente soportados por todas las implementaciones de *HTTP/1.0*. Éstos son: *PUT*, *LINK*, *UNLINK* y DELETE (que no estudiaremos ni trataremos en nuestro servidor).
- **URI** = *Universal Resource Identifier, o Uniform Resource Locator* (*URL*). El *URL* puede ser parcial, es decir no incluir todo el URL (por ejemplo cuando ya hemos accedido a la página principal de un web y hacemos referencia a un subdirectorio dentro del mismo) y no debe contener espacios en blanco.
- **VersionDelProtocolo** = *HTTP/1.0* (por ejemplo)
- **#13#10** = Retorno de Carro y Salto de línea

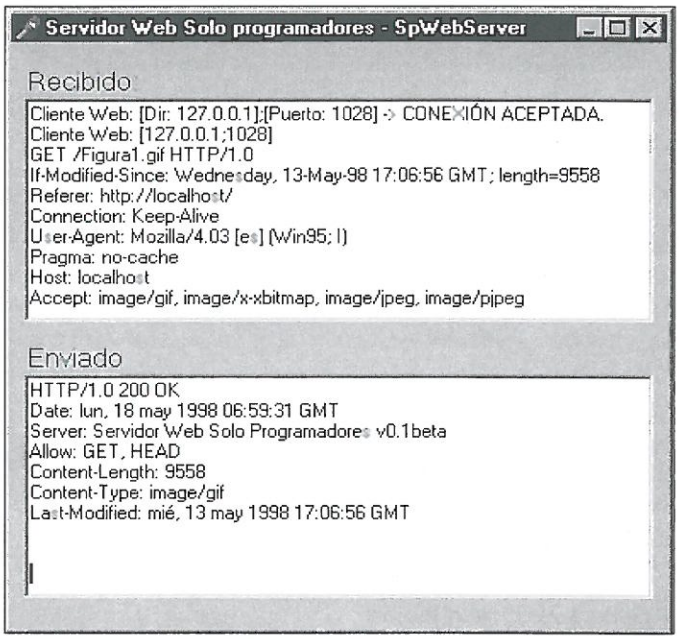


Figura 1:
Imagen del servidor web en funcionamiento. En la parte superior se muestran las peticiones recibidas y en la inferior, las cabeceras que envía el servidor

- **<Cabecera>** = <NombreDelCampo> : <Valor> #13#10
- **<datos>** = mensaje de acuerdo a la norma *MIME (Multipurpose Internet Mail Extensions)* definida primeramente en la **rfc1341** y redefinida en la **rfc1521.**

Al recibir una petición *HTTP* de un navegador, deberemos analizar el contenido de la misma para extraer el método de petición, la versión del protocolo, las distintas cabeceras (si queremos) y por último los datos. No detallaremos las distintas cabeceras de una petición, aunque nuestro servidor podría hacer un análisis de las mismas y presentar distintas respuestas de acuerdo con ellas. Por ejemplo, se podría querer enviar páginas diferentes a un cliente que utilice el Navegador Netscape de que a un cliente que use Internet Explorer.

Nuestro servidor sólo tratará (por ahora) los métodos de petición *GET* y *HEAD*.

Veamos por separado ambos métodos:

1. **Método GET**
   El método *GET* indica al servidor que el cliente quiere toda la información contenida en la *URI* enviada. Si la *URI* se refiere a un proceso (por ejemplo, a una secuencia *CGI*) se devolverán los datos que devuelva dicho proceso y no el propio proceso.

   El método *GET* consta de la palabra **GET** seguida del *URI* y de la versión del protocolo. Dos ejemplos que hacen referencia a una misma página son:

```
GET http://www.towercom.es/soloprog.html
    HTTP/1.0 #13#10
GET /soloprog.html HTTP/1.0 #13#10
```

   Para ampliar información el lector puede consultar la especificación **rfc1945.**

2. **Método HEAD**
   El método *HEAD* es igual que el método *GET*, pero en este caso el servidor web sólo debe enviar información de cabeceras (que serán idénticas a las enviadas a una petición con *GET*). Es útil para comprobar la validez de los hiperenlaces de una página. Por tanto, el método *GET* es similar al *HEAD*:

```
HEAD http://www.towercom.es/soloprog.html
    HTTP/1.0 #13#10
HEAD /soloprog.html HTTP/1.0 #13#10
```

Listado 1. Código completo del servidor web (y servidor de echo).

```
unit SpWebServer_Unit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ScktComp, StdCtrls;

type
  TForm1 = class(TForm)
    EchoServerSocket: TServerSocket;
    WebServerSocket: TServerSocket;
    MemoRecibido: TMemo;
    LabelRecibido: TLabel;
    LabelEnviado: TLabel;
    MemoEnviado: TMemo;
    procedure FormCreate(Sender: TObject);

    procedure EchoServerSocketAccept(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure EchoServerSocketClientRead(Sender: TObject;
      Socket: TCustomWinSocket);

    procedure WebServerSocketAccept(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure WebServerSocketClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure WebServerSocketClientWrite(Sender: TObject;
      Socket: TCustomWinSocket);
  private
    { Nombre del fichero solicitado en la petición y Handle para tratarlo }
    FicheroSolicitado: String;
    HandleFicheroSolicitado: File of Byte;
    FicheroAbierto: boolean;

    { Página y Directorio Inicial ("Home") }
    PaginaInicial: String;
    DirectorioInicial: String;

    Buffer: array[0..10000] of Char;

    { Función que devuelve la parte del string "AString" que está antes
      del primer espacio en blanco y deja en "AString" la segunda parte
      de dicho string }
    function  DevuelvePrimeraPalabra (var AString: String): String;

    { Procedimiento que vacía un Buffer }
    procedure LimpiarBuffer (Buffer: Pchar; Longitud: Integer);

    procedure EnviarCabeceras (Socket: TCustomWinSocket; CodigoHTTP: Integer);
    procedure EnviarFichero   (Socket: TCustomWinSocket);
    procedure CerrarSocket    (Socket: TCustomWinSocket);
  end;

var
  Form1: TForm1;

implementation
```

RESPUESTAS HTTP

En cuanto a las respuestas, las analizaremos con más detalle ya que son las que realmente nos interesan. La razón es que será esta información la que deba generar nuestro servidor. Recordamos que la versión *HTTP/0.9* definía *Respuestas Simples*, aunque nosotros no vayamos a considerarlas. Las que nos interesan son las *Respuestas Completas*, definidas a partir de la versión *HTTP/1.0*. Cuando hablemos de respuestas, nos referiremos siempre a respuestas completas.

De acuerdo con la especificación, la notación de un método de respuesta *HTTP* es la siguiente:

```
Línea de Estado  *(Cabecera General | Cabecera
      de Respuesta | Cabecera de Entidad)
      #13#10 [Cuerpo de la entidad]
```

- **Línea de Estado** = Consta de la versión del protocolo (*HTTP/1.0*), un espacio en blanco, un *Código de Estado* (ver **Tabla 1**), un espacio, una frase explicativa del código anterior y por último un retorno de carro y un salto de línea. Por ejemplo:

```
HTTP/1.0 404 Not Found #13#10
```

Los Códigos de Estado tienen siempre tres dígitos y pertenecen a uno de entre cinco grupos de códigos, cada uno de los cuales se distingue del resto a través del primer dígito. Este primer dígito define la clase de la respuesta:

- **1xx**: Informativo. No se usa (está reservado para usos futuros)
- **2xx**: Éxito. Indica que la petición se aceptó y se trató correctamente.
- **3xx**: Redirección. Informa que se ha de tomar otra acción para finalizar la petición.
- **4xx**: Error en el lado cliente. Debido a una mala sintaxis o a que el recurso solicitado no existe en el servidor.

- **5xx**: Error en el lado servidor. Provocado por un incompleto o erróneo funcionamiento del servidor.

- **Cabecera General** = cabecera aplicable tanto a peticiones como a respuestas y que no tiene relación con el cuerpo de la entidad sino con el propio mensaje transmitido. Sólo hay dos posibles cabeceras generales: **Date** *y* **Pragma**.

  - **Date**: fecha y hora en la que se ha originado el mensaje (en este caso, enviado al cliente). Por ejemplo: **Expires: dom, 31 May 1998 12:30:00 GMT.**
  - **Pragma**: se utiliza para incluir directivas de implementación aplicables a cualquier ordenador intermedio que reciba los mensajes de petición o respuesta. Estas cabeceras debe pasarlas cualquier *proxy o gateway* aunque no tengan sentido para ellos, pues pueden aplicarse a otros nodos del camino desde el cliente al servidor. El formato es el siguiente: **Pragma: no-cache | elemento [= palabra]**. La directriz **no-cache** indica a un sistema intermedio, que la petición debe enviarse al servidor, incluso aunque se tenga una copia de la misma en la caché.

- **Cabecera de Respuesta** = Permite pasar información adicional que no se puede incluir en la Línea de Estado. Hay tres tipos: **Location**, **Server** *y* **WWW-Authenticate**.

  - **Server**: línea en la que se informa del servidor web que se está utilizando. Por ejemplo: **Server: Internet Explorer/4.0.**
  - **Location**: localización exacta del recurso que hemos solicitado. Por ejemplo: **Location:** http://www.towercom.es/soloprog.html".
  - **WWW-Authenticate**: se utiliza en mensajes **401**, es decir cuan-

Listado 1. Continuación.

```
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  PaginaInicial := '\home.htm';
  DirectorioInicial := 'c:\home';
  FicheroAbierto := False;
end;

{-------------------------------------------------------------------}
{ Servidor echo }
procedure TForm1.EchoServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  MemoRecibido.Lines.Add
    ('Cliente echo: [Dir: ' + Socket.RemoteAddress +
     '];[Puerto: ' + IntToStr(Socket.RemotePort) +
     '] -> CONEXIÓN ACEPTADA.');
end;

procedure TForm1.EchoServerSocketClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
var Recibido: string;
begin
  Recibido := Socket.ReceiveText;
  MemoRecibido.Lines.Add ('Texto recibido: ' + Recibido);
  Socket.SendText (Recibido);
end;
{ Fin Servidor echo }
{-------------------------------------------------------------------}
{ SERVIDOR WEB }

procedure TForm1.WebServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  MemoRecibido.Lines.Add
    ('Cliente Web: [Dir: ' + Socket.RemoteAddress +
     '];[Puerto: ' + IntToStr(Socket.RemotePort) +
     '] -> CONEXIÓN ACEPTADA.');
end;

procedure TForm1.WebServerSocketClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
  Recibido: string;
  Comando: string;
  CodHTTP, i: Integer;
begin
  Recibido := Socket.ReceiveText;
  if (Recibido = '') then exit;
  MemoRecibido.Lines.Add ('Cliente Web: [' + Socket.RemoteAddress +
                          ';' + IntToStr(Socket.RemotePort) + ']');
  MemoRecibido.Lines.Add (Recibido);

  { Extraemos de la petición el comando (si existe) }
  Comando := DevuelvePrimeraPalabra (Recibido);
```

Listado 1. Continuación.

```
{ Si es una petición HTTP/1.0 (ó HTTP/1.1) buscamos el nombre del
  fichero pedido; en caso contrario será una petición HTTP/0.9 o
  la petición recibida será errónea: en ambos casos la ignoramos }
if (Pos('HTTP/1.0', Recibido) > 0) or (Pos('HTTP/1.1', Recibido) > 0) then
  FicheroSolicitado := DevuelvePrimeraPalabra (Recibido)
else begin
  CerrarSocket (Socket);
  exit;
end;

{ Cambiamos las barras que indican subcarpetas en Internet, por las
  equivalentes en MS-DOS y WINDOWS }
For i:= 1 to Length(FicheroSolicitado) do
  if (FicheroSolicitado[i] = '/') then FicheroSolicitado[i] := '\';

{ Comprobamos si se pidió la página por defecto o una ruta completa }
if (FicheroSolicitado = '\') then
  FicheroSolicitado := DirectorioInicial + PaginaInicial
else
  FicheroSolicitado := DirectorioInicial + FicheroSolicitado;

{ Si el comando es GET o HEAD y el fichero existe,
  y asignamos al código HTTP el número 200 -OK- }
if (Comando = 'GET') or (Comando = 'HEAD') then begin
  if FileExists(FicheroSolicitado) then begin
     AssignFile(HandleFicheroSolicitado, FicheroSolicitado);
     Reset(HandleFicheroSolicitado);
     FicheroAbierto := True;
     CodHTTP := 200
  end
  else
     CodHTTP := 404;
end
{ si el comando es POST, el código será "501, No implementado" y
  si no, el código será "400, Petición errónea" }
else if (Comando = 'POST') then
     CodHTTP := 501
else
     CodHTTP := 400;

{ Sea el comando que sea, enviamos las cabeceras de la respuesta HTTP }
EnviarCabeceras (Socket, CodHTTP);

{ Si el comando es GET y el fichero existe, abrimos el
  fichero -asignándolo al Handle del fichero-, y lo enviamos }
if (Comando = 'GET') and (CodHTTP = 200) then
  EnviarFichero (Socket)
else
  CerrarSocket (Socket);
end;

procedure TForm1.EnviarCabeceras (Socket: TCustomWinSocket; CodigoHTTP: Integer);
var
  Mensaje: string;
  TipoMIME: string;
  Extension: string;
  i: Integer;
begin
```

do se necesita autorización. Indica los esquemas de autenticación en el servidor.

- **Cabecera de Entidad** = Define información opcional sobre el *Cuerpo de Entidad* o sobre el recurso. Hay siete: **Allow, Content-Encoding, Content-Length, Content-Type, Expires, Last-Modified y Extension-header.**
  - **Allow**: Informa del conjunto de métodos de petición que soporta el servidor. Por ejemplo: **Allow: GET, HEAD.** Si el cliente intenta utilizar algún método no listado en *Allow*, el servidor le responderá con una respuesta de error (**501, No implementado**).
  - **Content-Encoding**: codificación utilizada en el recurso, de forma que el cliente pueda descodificarlo para obtener el tipo de datos definido en **Content-Type**. Por ejemplo: **Content-Encoding: x-gzip.**
  - **Content-Length**: tamaño del cuerpo de la entidad. Por ejemplo: **Content-Length: 4321.** Es muy útil para que el cliente sepa la información que va a recibir y de esta manera, reserve un tamaño de *buffer* apropiado.
  - **Content-Type**: indica el tipo de los datos del cuerpo de la entidad. Por ejemplo: **Content-Type: text/html.**
  - **Expires**: fecha y hora después de la cual se debe recoger el recurso del servidor en lugar de cogerlo de la caché (si está en ella). Por ejemplo: **Expires: lun, 04 Feb 1998 04:00:00 GMT.**
  - **Last-Modified**: fecha y hora en la que el recurso se modificó en el servidor por última vez. Por ejemplo: **Last-Modified: lun, 04 Feb 1998 05:00:00 GMT.**
  - **Extension-Header**: cabecera *HTTP*.
- **Cuerpo de Entidad** = Puede o no existir, dependiendo de la petición.

Contiene los datos solicitados por el cliente, en forma de una secuencia de octetos.

LOS PRIMEROS PASOS

Vistas las cabeceras que debe (o puede) generar un servidor, veremos cómo se debe realizar su implementación. Aprovecharemos el servidor de *echo* que desarrollamos en el artículo del mes pasado (y cuyo código dejaremos intacto) y en él añadiremos otro componente *TserverSocket* que, a la postre, será el núcleo del servidor web.

Para ello, dejamos caer desde la paleta de componentes (pestaña *Internet*) un componente *TServerSocket* en el formulario de la unidad. A este componente lo llamaremos **WebServerSocket** y al que ya existía para el servicio de *echo* le llamaremos **EchoServerSocket**.

Añadiremos al **WebServerSocket** las propiedades que lo convertirán en un servidor del protocolo *HTTP*: el puerto (propiedad *Port*), será el **80** y el servicio (propiedad *Service*), lo definiremos como **http** (ojo, pues el servicio debe estar declarado en el fichero **Services** del directorio de Windows 95, ya que al prevalecer la propiedad *Service* sobre la propiedad *Port*, si aquélla no está definida, dejamos el *socket* servidor con un puerto indefinido). También fijaremos la propiedad *Active* a **True**, para que el Servidor se active nada más iniciar la aplicación.

Al igual que en el servidor de *echo* las peticiones que le se pueden investigar desde el evento *OnClientRead*. De igual manera, el evento *OnClientWrite* nos avisa que el *buffer* de envío de datos está vacío y por tanto preparado para que se le envíe información fresca. Ambos eventos llevan como parámetro, aparte del *Sender*, un objeto de la clase *TCustomWinSocket* que guarda toda la información relativa a la conexión del servidor con un cliente con-

Listado 1. Continuación.

```
{ Creamos el código de respuesta, tal y como viene definido en
  la especificación HTTP/1.0 - rfc1945 }
{ Primera línea, "Línea de estado" }
case CodigoHTTP of
   200: Mensaje := 'HTTP/1.0 200 OK' + #13#10;
   201: Mensaje := 'HTTP/1.0 201 Created' + #13#10;
   202: Mensaje := 'HTTP/1.0 202 Accepted' + #13#10;
   204: Mensaje := 'HTTP/1.0 204 No Content' + #13#10;
   301: Mensaje := 'HTTP/1.0 301 Moved Permanently' + #13#10;
   302: Mensaje := 'HTTP/1.0 302 Moved Temporarily' + #13#10;
   304: Mensaje := 'HTTP/1.0 304 Not Modified' + #13#10;
   400: Mensaje := 'HTTP/1.0 400 Bad Request' + #13#10;
   401: Mensaje := 'HTTP/1.0 401 Unauthorized' + #13#10;
   403: Mensaje := 'HTTP/1.0 403 Forbidden' + #13#10;
   404: Mensaje := 'HTTP/1.0 404 Not Found' + #13#10;
   500: Mensaje := 'HTTP/1.0 500 Internal Server Error' + #13#10;
   501: Mensaje := 'HTTP/1.0 501 Not Implemented' + #13#10;
   502: Mensaje := 'HTTP/1.0 502 Bad Gateway' + #13#10;
   503: Mensaje := 'HTTP/1.0 503 Service Unavailable' + #13#10;
end;

{ Siguiente línea (Cabecera General): Fecha y hora }
Mensaje := Mensaje + 'Date: ' +
        FormatDateTime('ddd, d mmm yyyy hh:mm:ss "GMT"', Now()) + #13#10;

{ Siguiente línea (Cabecera de Respuesta): Nombre de nuestro Servidor }
Mensaje := Mensaje + 'Server: Servidor Web Solo Programadores v0.1beta' + #13#10;

{ Enviamos las cabeceras de contenido, salvo que no exista el recurso, en
  cuyo caso enviamos el nombre del fichero }
if CodigoHTTP = 200 then begin
  { Comandos de petición permitidos }
  Mensaje := Mensaje + 'Allow: GET, HEAD' + #13#10;

   { Content-Length }
  Mensaje := Mensaje + 'Content-Length: ' +
          IntToStr(FileSize(HandleFicheroSolicitado)) + #13#10;

  { Escribimos el tipo de fichero por defecto (html) pero buscamos otras
    extensiones: aquí podremos codificar tantas como tipos MIME }
  TipoMIME := 'text/html';
  i := Pos('.', FicheroSolicitado);
  if (i > 0) then begin
     Extension := Copy(FicheroSolicitado, (i + 1), 3);
     if     (UpperCase(Extension) = 'GIF') then TipoMIME := 'image/gif'
     else if (UpperCase(Extension) = 'JPG') then TipoMIME := 'image/jpeg'
     else if (UpperCase(Extension) = 'TXT') then TipoMIME := 'text/plain';
  end;

  { Content-Type }
  Mensaje := Mensaje + 'Content-Type: ' + TipoMIME + #13#10;

  { Last-Modified }
  Mensaje := Mensaje + 'Last-Modified: ' +
          FormatDateTime('ddd, d mmm yyyy hh:mm:ss "GMT"',
          FileDateToDateTime (FileAge(FicheroSolicitado))) + #13#10;

  { Línea en blanco }
```

Listado 1. Continuación.

```
    Mensaje := Mensaje + #13#10;
  end;

  { Enviamos todo }
  Socket.SendText (Mensaje);
  { y lo mostramos en pantalla }
  MemoEnviado.Lines.Add (Mensaje);
end;

procedure TForm1.EnviarFichero (Socket: TCustomWinSocket);
var
  ByteNum: Integer;
  ByteLeido: Byte;
begin
  { Queremos que nos avise cuando hay datos nuevos o cuando el buffer de
    transmisión se ha vaciado o cuando la conexión ha terminado }
  Socket.AsyncStyles := [asRead, asWrite, asClose];
  if FicheroAbierto then begin

    { Si no hemos llegado al fin del fichero, limpiamos el Buffer de escritura
      y copiamos desde el fichero al Buffer. Y luego enviamos todo }
    if not Socket.Connected then
      CerrarSocket (Socket)
    else if Eof(HandleFicheroSolicitado) then
      CerrarSocket (Socket)
    else begin
      { Una vez que empezamos a enviar el fichero, sólo queremos que
        nos avise cuando se vacíe el buffer de transmisión o cuando
        la conexión haya terminado }
      Socket.AsyncStyles := [asWrite, asClose];
      ByteNum := 0;

      { Limpiamos el Buffer }
      LimpiarBuffer (Buffer, 10000);

      { Copiamos el fichero en el Buffer }
      while ((ByteNum < 5000) and (not Eof(HandleFicheroSolicitado))) do begin
        Read(HandleFicheroSolicitado, ByteLeido);
        Buffer[ByteNum] := Char(ByteLeido);
        Inc(ByteNum);
      end;
      Socket.SendBuf (Buffer, ByteNum);
    end;
  end;
end;

procedure TForm1.WebServerSocketClientWrite(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  EnviarFichero (Socket);
end;

procedure TForm1.CerrarSocket (Socket: TCustomWinSocket);
begin
  if FicheroAbierto then begin
    CloseFile (HandleFicheroSolicitado);
    FicheroAbierto := False;
  end;
```

creto. Este parámetro viene definido como **Socket: TcustomWinSocket**, aunque en realidad es de la clase *TserverClientWinSocket*, heredera de forma directa de *TCustomWinSocket*. Este parámetro, además de recibir información, permitirá utilizar todos los métodos asociados a su clase, entre los que se encuentran los de envío y recepción de datos.

Lo primero que haremos será definir un manejador de ambos eventos, para lo cual haremos doble clic en las filas correspondientes de la pestaña *Events* (Eventos) del *Object Inspector*, lo cual generará el siguiente código:

```
Procedure
      TForm1.EchoServerSocketClientRead(Send
      er: TObject; Socket: TCustomWinSocket);
begin
end;

procedure
      TForm1.EchoServerSocketClientWrite(Send
      er: TObject; Socket: TCustomWinSocket);
begin
end;
```

Realizamos un inciso. En el servidor habrá que definir un directorio a partir del cual se podrán referenciar los recursos en las peticiones *HTTP*. Si, por ejemplo, este directorio fuese **C:\WEB\HOME** y nuestro nombre de dominio fuera www.towercom.es, las peticiones dirigidas a http://www.towercom.es/soloprog/soloprog.htm habrá que dirigirlas al directorio local: **C:\WEB\HOME\soloprog\soloprog.htm**.

Además, dado que el servidor web puede recibir peticiones en las que en el URL sólo se especifique el nombre del servidor, pero no el recurso solicitado, habrá que definir una página HTML por defecto. De esta forma si la página por defecto es **C:\WEB\HOME\home.htm**, todas las peticiones realizadas a: http://www.towercom.es recibirán dicha página.

Para definir estos dos parámetros se declaran dos variables (públicas):

```
PaginaInicial: String;
DirectorioInicial: String;
```

Y se inicializan en el evento *OnCreate* del formulario:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  PaginaInicial := '\home.htm';
  DirectorioInicial := 'c:\home';
end;
```

MANEJADOR DEL EVENTO ONCLIENTREAD

Comenzamos ahora con el manejador del evento *OnClientRead*. Como lo primero que llega de una petición HTTP son una serie de cabeceras de texto, emplearemos el método *ReceiveText* del parámetro *Socket*, para obtener estas cabeceras. El texto recibido lo asignaremos a una variable de tipo *string* que llamaremos: **Recibido**. Lo primero que haremos será comprobar que esta cadena no ha llegado vacía, y en caso de que llegue vacía la ignoraremos.

```
Recibido := Socket.ReceiveText;
if (Recibido = '') then exit;
```

Añadiremos el texto recibido al *Memo* desde el que vamos a ir monitorizando las peticiones de los clientes. A este *Memo* le hemos llamado **MemoRecibido** y hemos creado un nuevo *Memo*, **MemoEnviado** para escribir las cabeceras que genera el servidor (ver Figura 1). Como todas las cabeceras terminan en *#13#10*, los saltos de línea, en el **MemoRecibido**, se insertan de forma automática. El código será:

```
MemoRecibido.Lines.Add ('Cliente Web: [' +
      Socket.RemoteAddress + ';' +
      IntToStr(Socket.RemotePort) + ']');
MemoRecibido.Lines.Add (Recibido);
```

En la primera línea de la petición viene el comando, el URL solicitado y la versión del protocolo. Extraemos el comando de petición, para lo cual creamos una función de uso general que extrae la primera palabra de un *string* y devuelve el resto del *string*. Esta función la hemos denominado **DevuelvePrimeraPalabra** y se puede observar en el Listado 1 (como función general, su explicación queda fuera del objetivo de este artículo).

Posteriormente comprobamos la versión *HTTP* que utiliza el cliente, buscando en la petición el *string*: **HTTP/1.0** ó **HTTP/1.1**. Si la versión es la 1.0 o la 1.1, extraemos el nombre del recurso solicitado (en la variable **FicheroSolicitado**) y en caso contrario cerramos la conexión:

```
Comando := DevuelvePrimeraPalabra (Recibido);
if (Pos('HTTP/1.0', Recibido) > 0) or
      (Pos('HTTP/1.1', Recibido) > 0) then
  FicheroSolicitado := DevuelvePrimeraPalabra
      (Recibido)
else begin
  CerrarSocket (Socket);
  exit;
end;
```

Si la versión de la petición *HTTP* no es la 0.9, cambiamos las barras que indican subcarpetas en Internet, por las barras de directorios del DOS-WINDOWS. Si el recurso solicitado es la página por defecto, el **FicheroSolicitado** será la **PaginaInicial** y en caso contrario, será la página obtenida al añadir al **DirectorioInicial** el **FicheroSolicitado**.

Listado 1. Continuación.

```
  Socket.Close;
end;

procedure TForm1.LimpiarBuffer (Buffer: Pchar; Longitud: Integer);
var i: Integer;
begin
  For i:= 0 to Longitud - 1 do begin
    Buffer^ := Char(0);
    Inc(Buffer);
  end;
end;

function TForm1.DevuelvePrimeraPalabra (var AString: String): String;
var j, Longitud: Integer;
    Aux: String;
begin
  Aux := '';
  j := 1;
  Longitud := Length(AString);
  while (AString[j] <> ' ') and (j <= Longitud) do begin
    Aux := Aux + AString[j];
    Inc(j);
  end;

  { Devuelve la primera parte del string, antes del 1er espacio }
  Result := Aux;

  { Comprobamos si se ha llegado al final de la línea y en caso
    contrario, se deja en AString la segunda parte del string }
  if (j > Longitud) then
    AString := ''
  else
    Delete(AString, 1, j);
end;

end.
```

```
For i:= 1 to Length(FicheroSolicitado) do
 if (FicheroSolicitado[i] = '/') then
      FicheroSolicitado[i] := '\';
if (FicheroSolicitado = '\') then
 FicheroSolicitado := DirectorioInicial +
      PaginaInicial
else
 FicheroSolicitado := DirectorioInicial +
      FicheroSolicitado;
```

A continuación analizaremos el comando. Si el comando es *GET* o *HEAD*, y el fichero existe en el disco de nuestro servidor, el código *HTTP* que hay que enviarle al cliente es **"200, OK"** (todo correcto). Si el fichero no existe, se le enviará el código: **"404: Not Found"** (no encontrado).

Si el fichero existe, se abre y se utiliza una variable *booleana* **FicheroAbierto** para informar si el fichero está abierto o ya se ha cerrado (muy importante, pues para enviar un fichero se debe ir llenando un *buffer* de datos, que será generalmente de menor tamaño que el propio fichero).

Si el comando es *HEAD*, debemos enviar una respuesta con código **501**, que informa al cliente que nuestro servidor no tiene implementado el soporte a ese comando. Para cualquier otro comando distinto de los tres anteriores, el código *HTTP* será: **"400: Bad Request"** (petición errónea).

```
if (Comando = 'GET') or (Comando = 'HEAD')
      then begin
 if FileExists(FicheroSolicitado) then begin
  AssignFile(HandleFicheroSolicitado,
      FicheroSolicitado);
  Reset(HandleFicheroSolicitado);
  FicheroAbierto := True;
  CodHTTP := 200;
 end
 else
  CodHTTP := 404;
end
else if (Comando = 'POST') then
 CodHTTP := 501
else
 CodHTTP := 400;
```

Cualquiera que sea el comando recibido (incluso los comandos no reconocidos) el servidor debe generar una respuesta con una serie de cabeceras, de acuerdo a la especificación. Sólo en el caso de que la petición sea *GET* y el recurso exista, se enviará además al cliente el recurso solicitado; en cualquier otro caso, tras enviar las cabeceras, se cerrará la conexión con el *socket* cliente:

```
EnviarCabeceras (Socket, CodHTTP);
if (Comando = 'GET') and (CodHTTP = 200) then
 EnviarFichero (Socket)
else
 CerrarSocket (Socket);
```

Como se puede observar en el código anterior, se han definido dos procedimientos de "envío": uno para mandar las cabeceras *HTTP* (**EnviarCabeceras**) -que toma como parámetros el *socket* servidor y el código *HTTP* de la respuesta- y otro para enviar el recurso solicitado (**EnviarFichero**) –que toma como único parámetro el *socket* servidor que dialoga con el cliente en cuestión-.

En cuanto a la función **CerrarSocket**, ésta se encarga de cerrar la conexión del *socket* servidor y de cerrar el fichero, si está abierto (ver Listado 1).

MANEJADOR DEL EVENTO ONCLIENTWRITE

Este evento se dispara cuando el *buffer* de datos de salida está vacío, y por tanto, dispuesto para recibir nuevos datos. Nos servirá, por tanto, para saber cuándo podemos enviar nuevos datos, en caso de que quede información pendiente de transmitir.

En nuestro caso introduciremos aquí el código necesario para terminar de enviar un recurso (si el *buffer* de transmisión es menor que el tamaño del fichero). El código que se debe introducir será por tanto una sola línea:

```
Procedure
      TForm1.WebServerSocketClientWrite(Sende
      r: TObject; Socket: TCustomWinSocket);
begin
 EnviarFichero (Socket);
end;
```

PROCEDIMIENTO ENVIARCABECERAS

Este procedimiento toma como parámetros el *socket* servidor y el código de la respuesta *HTTP* y genera todas las cabeceras de dicha respuesta. La primera cabecera es la *Línea de Estado*, que se obtiene directamente del código *HTTP* pasado como parámetro (ver Listado 1).

Después se envía la *Cabecera General* **Date**, recogiendo la fecha y hora local, con la función de Delphi **Now** y formateándola de acuerdo al formato general de fechas (Listado 1).

La siguiente cabecera elegida es la *Cabecera de Respuesta* **Server**, en la que escribimos el nombre que queramos dar a nuestro servidor web y la versión del mismo (Listado 1).

A continuación vamos a enviar cuatro *Cabeceras de Entidad*: **Allow, Content-Length, Content-Type y Last-Modified**, seguidas de una línea en blanco, como requiere la especificación del protocolo *HTTP*; en *Allow* permitimos sólo los métodos *GET* y *HEAD*; en *Content-Length* la longitud del recurso solicitado; en *Content-Type* escribimos el tipo MIME de dicho recurso; y para finalizar, en *Last-Modified*, la fecha y hora en que fue modificado por última vez (Listado 1).

Una vez construido el mensaje, se lo enviamos al cliente y lo mostramos en pantalla, para monitorizar la salida con el siguiente paso:

```
Socket.SendText (Mensaje);
MemoEnviado.Lines.Add (Mensaje);
```

PROCEDIMIENTO ENVIARFICHERO

Este procedimiento sólo será utilizado cuando se haya recibido una petición con el comando *GET* y tengamos abierto un fichero. En ese caso, se enviará a través del *socket* el contenido del fichero. Para conseguirlo se copiará la información desde el fichero en cuestión a un *buffer* y se enviará esta información mediante la función *SendBuf* de la clase *TServerClientWinSocket*.

```
ByteNum := 0;
LimpiarBuffer (Buffer, 10000);
while ((ByteNum < 5000) and (not
      Eof(HandleFicheroSolicitado))) do begin
  Read(HandleFicheroSolicitado, ByteLeido);
  Buffer[ByteNum] := Char(ByteLeido);
  Inc(ByteNum);
end;
Socket.SendBuf (Buffer, ByteNum);
```

El procedimiento LimpiarBuffer se encarga de vaciar el *buffer* de salida y se puede observar en el Listado 1.

Existe una propiedad que hay que establecer en el *socket* servidor y es la de *AsyncStyles*. Esta propiedad, como ya vimos en el anterior artículo, puede tener como valor: **asRead, asWrite, asAccept, asConnect y asClose**. Nos interesa que después de recibir la primera conexión, nos avise ante cualquier evento de lectura, escritura o de finalización de la conexión:

```
Socket.AsyncStyles := [asRead, asWrite, asClose];
```

Sin embargo, al comenzar a enviar el fichero, sólo nos interesa que genere un evento cuando el *buffer* de salida está dispuesto para que enviemos nuevos datos o cuando haya concluido la conexión:

Tabla 1: Códigos de estado de una respuesta, frase explicativa "estándar" y significado de cada uno de ellos.

CÓDIGO ESTADO	FRASE EXPLICATIVA DEL CÓDIGO	SIGNIFICADO
200	*OK*	Todo Correcto
201	*Created*	Petición atendida, pero todavía no aceptada
202	*Accepted*	Petición creada y aceptada, pero todavía pendiente de ser procesada
204	*No Content*	La petición ha sido aceptada, pero no hay información nueva
301	*Moved Permanently*	El URL pedido ha cambiado de sitio
302	*Moved Temporarily*	El URL pedido ha cambiado de sitio temporalmente
304	*Not Modified*	Respuesta a una petición condicional a que haya cambiado el recurso
400	*Bad Request*	La sintaxis de la petición es incorrecta
401	*Unauthorized*	El recurso solicitado requiere una contraseña
403	*Forbidden*	El recurso no tiene permitido el acceso
404	*Not Found*	Recurso no encontrado en el servidor
500	*Internal Server Error*	Error del propio servidor
501	*Not Implemented*	Característica no implementada en el servidor
502	*Bad Gateway*	El servidor, actuando como proxy, ha recibido una respuesta invalida de un servidor
503	*Service Unavailable*	El servidor está sobrecargado o está temporalmente desactivado (en mantenimiento)

```
Socket.AsyncStyles := [asWrite, asClose];
```

Y con esto, tenemos terminado el servidor web. Se puede ver el código completo en el Listado 1.

Se incluye en el CD-ROM de la revista todo este código y el ejecutable del servidor.

También se incluye el cliente *echo* que creamos en el anterior artículo, con el objeto de que se puedan probar los dos servicios disponibles en nuestro servidor: HTTP y *echo*.

Por último se incluye una página Web (Home.htm) y una imagen (Figura1.gif) que se pueden utilizar como página inicial del Servidor (siempre y cuando se copien en un directorio llamado **C:\Home**.

Para probar el servidor web, basta con tener un navegador (Netscape o Internet Explorer) y en el propio PC lanzar una petición al *host local* (nuestro PC), mediante cualquiera de las URL's siguientes:

http://localhost
http://localhost/Home.htm

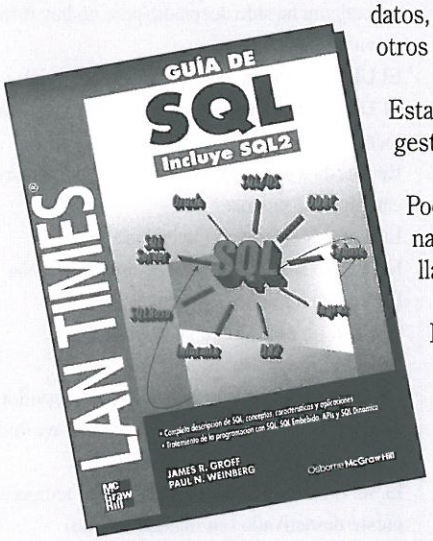
CONCLUSIÓN

Hemos visto cómo crear un servidor web, apreciando la rapidez y facilidad con la que se generan aplicaciones de este tipo en el entorno de desarrollo de Delphi. En el próximo artículo ampliaremos la funcionalidad del servidor, para que dé soporte a métodos *POST* y de esta manera pueda dialogar con secuencias *CGI*. Conseguiremos así un diálogo bidireccional en el que ya no sólo es el servidor quien envía datos al cliente, sino también el cliente puede enviar datos y actualizar registros en Bases de Datos residentes en el Servidor.

LIBROS

La Guía LAN Times de SQL

La guía LAN Times de SQL es la guía completa de SQL, el lenguaje estructurado de consultas. Describe el lenguaje de base de datos especificado por el estándar ANSI/ISO SQL y utilizado actualmente por los principales sistemas de gestión de bases de datos, incluyendo SQL Server, Oracle, Sybase, DB2, Ingres, SQL/DS, VAX SQL, SQLBase y otros muchos.



Esta guía le ayudará a entender fácilmente los conceptos de SQL y el papel de SQL en la gestión de bases de datos.

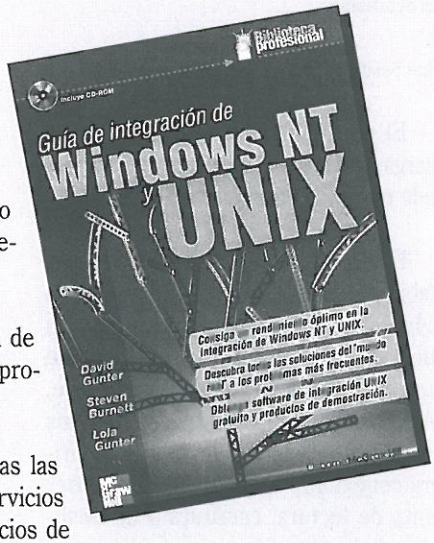
Podrá utilizar esta herramienta para recuperar y almacenar datos de ordenadores personales, redes de área local y mainframes. Además incluye numerosos ejemplos desarrollados paso a paso.

Independientemente del producto SQL que se esté utilizando, la Guí LAN Times de SQL ofrece una única perspectiva de SQL y de cómo utilizarlo para sacar el mayor rendimiento posible de la manera más sencilla.

Editorial: Mc Graw Hill
Nº páginas: 631
Nivel: Todos los niveles

Autor: James R.Groff y Paul N.Weinberg
Idioma: Español
Precio: 5.000 Ptas.(I.V.A. inc.)

Guía de Integración de Windows NT UNIX



Si desea aprender a administrar una red NT/UNIX para obtener el mejor rendimiento y prestaciones, puede echar mano de este libro escrito por un equipo de autores líderes y expertos en sistemas.

El libro le proporcionará una completa cobertura sobre el tema de integración de sistemas Windows NT y UNIX en el mundo real, ofreciendo desde soluciones a los problemas más frecuentes hasta técnicas para la administración de red más efectiva.

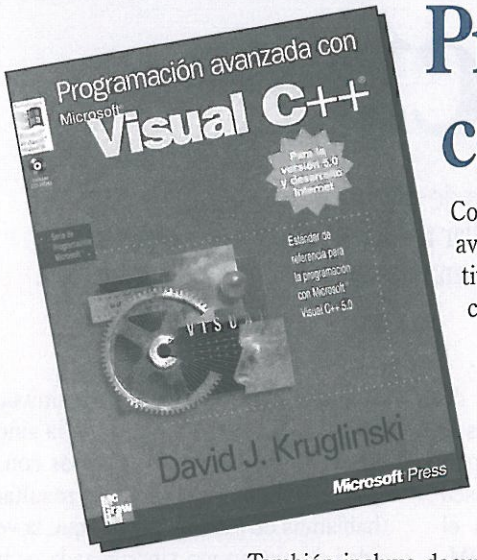
Junto con el CD-ROM que se incluye, el libro le ayudará a comprender todas las características de la integración de ambos sistemas, tratando temas como son los servicios de archivo en ambas plataformas, copias de seguridad en entornos integrados, servicios de correo, trabajo en redes con ambos sistemas, configuración de DNS, servicios de acceso remoto y aplicaciones de escritorio en plataformas cruzadas.

El CD-ROM que acompaña al libro incluye amplia información así como software gratuito.

Editorial: Mc Graw Hill
Nº páginas: 473
Nivel: Intermedio/avanzado

Autor: David Gunter, Steven Burnett, Lola Gunter
Idioma: Español
Precio: 5.500 Ptas.(I.V.A. inc.)

LIBROS



Programación avanzada con Visual C++

Construido sobre los sólidos logros de las tres ediciones anteriores, "Programación avanzada con Microsoft Visual C++" proporciona la cobertura detallada y exhaustiva del sistema de desarrollo Visual c++ y de las peculiaridades de la programación bajo el entorno del sistema operativo Microsoft Windows de 32 bits.

Este libro proporciona, sin lugar a dudas, la mejor referencia sobre las posibilidades de la compleja y potente herramienta de desarrollo de Microsoft.

El libro aporta información sobre fundamentos, gestión de eventos, GDI, cuadros de diálogo etc.

También incluye documentación sobre temas avanzados como programación mulithilo, DIB, ODBC y DLL; así como sobre creación de objetos documento y controles ActiveX, COM, automatización y utilización de asistentes.

Repasa también la programación en C++ para Internet, Winsock, Winlnet de la MFC y programas de ampliación de ISAPI

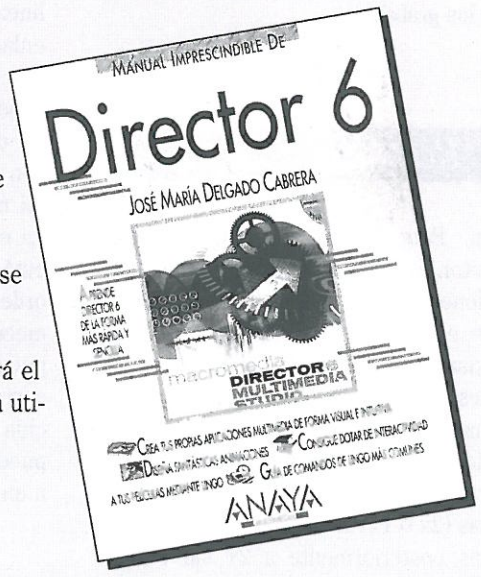
Al libro lo acompaña un CD-ROM que incluye todos los archivos de código fuente y las aplicaciones de ejemplo que conlleva el manual.

Editorial: Mc Graw Hill
Nº páginas: 931
Nivel: Avanzado

Autor: David J. Kruglisnski
Idioma: Español
Precio: 9.000 Ptas.(I.V.A. inc.)

Director 6

Director 6 es una aplicación para el desarrollo visual de cualquier proyecto multimedia y cuando es visual es porque se trata de una herramienta en la que la mayoría de las operaciones se pueden hacer de una forma rápida e intuitiva. Además de todo esto, Director 6 viene acompañado de un completo lenguaje de programación orientado a objetos denominado Lingo.



Con este libro adquirirá todos los conocimientos necesarios para manejarse con soltura dentro de un entorno como Director.

A lo largo de los catorce capítulos y dos apéndices de este libro encontrará el material imprescindible para desarrollar impresionantes aplicaciones multimedia utilizando esta potente herramienta.

Editorial: Anaya Multimedia
Nº páginas: 400
Nivel: Medio

Autor: José Mª Delgado Cabrera
Idioma: Español
Precio: 2.995 Ptas.(I.V.A. inc.)

Correo del lector

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relacionado con la programación.

Pregunta

¡Hola!. Poseo una grabadora de cds PHILIPS CDD3610 y tengo problemas a la hora de extraer audio de los Compact Discs de música. En el cd al que me refiero las canciones contienen ruido y pitidos muy molestos, como si saltara el reproductor de música. Me he dado cuenta que los ficheros wav que me crea el EasyCD-Creator también se escuchan igual, me gustaría saber cómo puedo solucionar esto, pues ni leyendo a menos velocidad consigo eliminarlos. También me gustaría preguntar por qué cuando utilizo cds regrabables, éstos no funcionan nada más que con la grabadora, aunque utilice el mismo procedimiento de grabación que con los grabables.

Michel Jiménez (Madrid)

Respuesta

Bien. Para los usuarios de EasyCD-Creator, hay una barra en el apartado opciones-utilidades-avanzadas que te permite enlentecer la extracción de audio precisamente para evitar esos *clicks* tan molestos que son fruto de una lectura demasiado rápida. También se puede solucionar extrayendo antes las canciones (para lo que aconsejamos usar velocidades lentas (2x ó 1x) al disco duro a fín de grabarlas posteriormente a 2x sin ningún problema.

En realidad, en la grabación de audio, la velocidad de grabación no es la que produce estos fallos, sino la velocidad de extracción de los datos del disco origen. La razón técnica estriba en el formato de estos CDs. En un CD-ROM, los datos están organizados en sectores. Cada sector tiene la dirección incorporada junto con los datos. Gracias a estos enlaces, una unidad de CD-ROM puede saltar directamente a la localización exacta de los datos, permitiendo a grandes velocidades, cuando el bus de datos del PC está ocupado, retornar al bloque exacto y continuar la lectura sin ningún error.

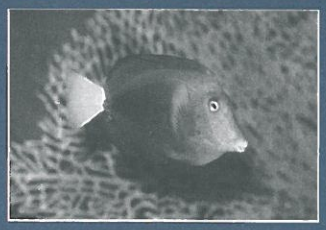
Los CDs de música son diferentes. Están diseñados para una reproducción lineal y no hay sectores o direcciones enlazados con los datos directamente. Sin embargo hay subcódigos, pero los subcódigos de información extraídos de los datos pueden no estar sincronizados con la información del bus de datos. De tal manera que si hay una interrupción en el trasiego de la información (el bus está ocupado, lo cual puede depender del ordenador, de qué programas se están ejecutando en ese momento), los datos pueden no ser adquiridos en el punto correcto donde se produjo la interrupción. Esto rompe el flujo de datos y puede generar ecos y saltos perfectamente audibles en la señal de audio.

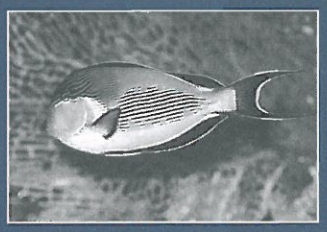
Existen ciertos programas denominados *rippers* para realizar estas extracciones de canciones de CDs de música a ficheros WAV que sí permiten la sincronización, y es con estos últimos con los que se obtienen los mejores resultados (hablamos de calidad, puesto que la velocidad de extracción sincronizada es bastante lenta (no suele superar 2.3x).

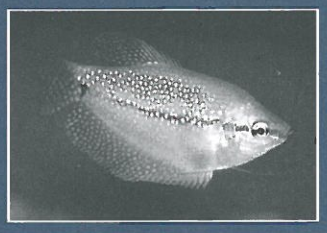
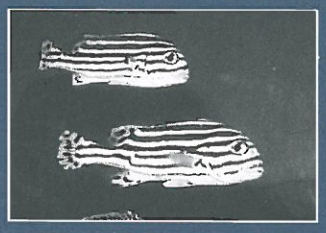
Y repondiendo ahora a tu pregunta de los CDs regrabables, estos sólo son reconocidos por regrabadoras como la tuya. Esto es porque un CD regrabable cambia de fase a diferencia de los CDs grabables una sóla vez. En su estado original, la capa de grabación del disco es policristalina, durante la grabación el láser convierte ciertas áreas a fase amorfa de manera que la reflectancia de estas áreas es mucho menor que las áreas cristalinas, consiguiendo así los ceros y unos de la información del disco.

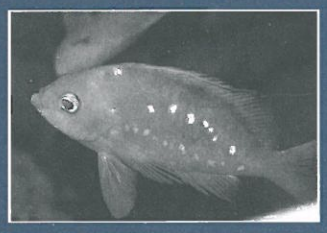
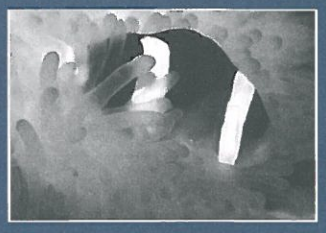
Durante la regrabación, algunas áreas amorfas de las pistas vuelven otra vez a su fase cristalina mediante el mismo láser pero a menor intensidad.

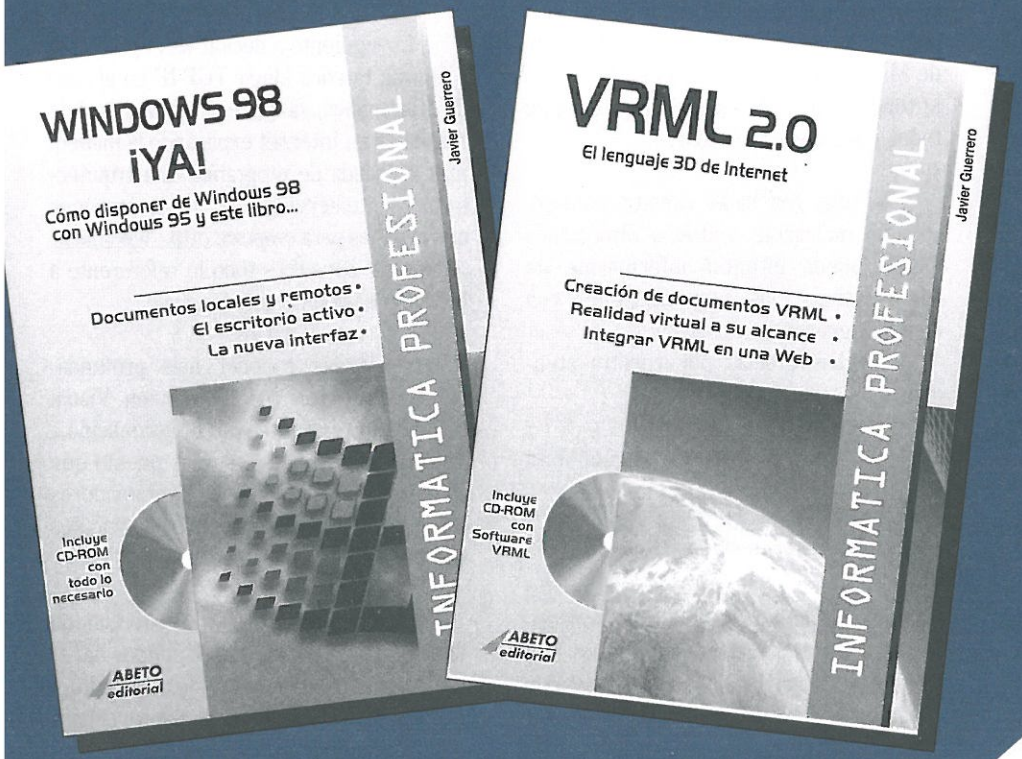
Los CDs grabables una sóla vez, usan la luz del láser para crear burbujas y deformaciones en el interior de la capa de grabación como partes amorfas. La reflectancia de un CD regrabable es menor que la de un CD estampado o grabable, así pues este primer tipo de CDs no son leíbles por la mayoría de las unidades de CD-ROM, pero sí por aquellas que soporten *"Multilectura"* como algunas unidades de CD-ROM y DVD-ROM.

PARA MOVERTE  EN

 INFORMÁTICA COMO

 PEZ EN 

EL  AGUA 



1995 ptas.
IVA incluido

Incluyen CD-ROM de regalo

Ya a la venta en quioscos y librerías.

ABETO editorial

c/ Aragoneses, 7 • 28108 Alcobendas (Madrid)
Tel.: (91) 661 42 11* • Fax: (91) 661 43 86

Pregunta

Buenas. Tengo una red de tres ordenadores a los que recientemente he añadido un cuarto un poco antiguo. Es un 486 de los primeros, con dos controladoras de discos IDE (puesto que tengo instalados una unidad lectora de CDROM y tres discos duros). La cuestión es que para poder poner el tercer disco en la tercera controladora, me dieron un disquete en la tienda con un driver que se carga en el MBR del primer disco. El acceso bajo MSDOS se realiza correctamente, pero mi interés es instalar Linux en este cuarto cliente puesto que la red la tengo bajo este S.O. Mi pregunta es la siguiente, ¿cómo puedo hacer que arranque Linux desde los discos duros (sin disquete) puesto que no puedo usar LILO?

Juan J. Carrascosa

Respuesta

La solución de tu problema es muy fácil. De hecho seguramente no te habrás percatado de ella antes como nos ha sucedido a todos en un principio. El gestor de arranque LILO de Linux, tiene una opción para instalarse en otra zona distinta del *Master Boot Record* o de un disquete. Escoge la opción de instalación *"en el superbloque de la partición Linux"* y no tendrás ningún problema con el driver del tercer disco, puesto que no toca el *Master Boot Record*. Eso sí, asegúrate después de configurar las particiones de arranque seleccionando como activa la partición de Linux para que Lilo se inicie al arrancar el ordenador.

Pregunta

Estimados amigos: estoy trabajando con la idea de realizar un pequeño juego de estrategia por turnos para varios jugadores (entre cinco y diez, más o menos). En una primera aproximación seguramente haré que cada jugador efectúe su turno para enviarlo en un fichero, vía disco, al ordenador "servidor". Este ejecutará los turnos de todos los jugadores y devolverá el resultado a cada jugador en un fichero, de nuevo vía disco. Obviamente esta solución es un tanto engorrosa y me gustaría poder hacer todas estas labores entre jugadores además vía módem.

El problema es: ¿qué lenguaje debo escoger teniendo esto en mente?. Conozco bien Turbo Pascal, C, C++, Visual Basic (hasta ahora sólo dispongo de la posibilidad de hacerme con la versión 4.0 de este último mediante la oferta de Microsoft para estudiantes), Java y no le tengo miedo al ensamblador. ¿Cuál de todos ellos me aconsejáis?.

Y, una vez dado vuestro consejo, ¿podéis indicarme textos o direcciones donde pueda intentar informarme de cómo manejar las comunicaciones con dicho lenguaje?.

Muchas gracias por vuestra atención.

Juan de Miguel Veira

Respuesta

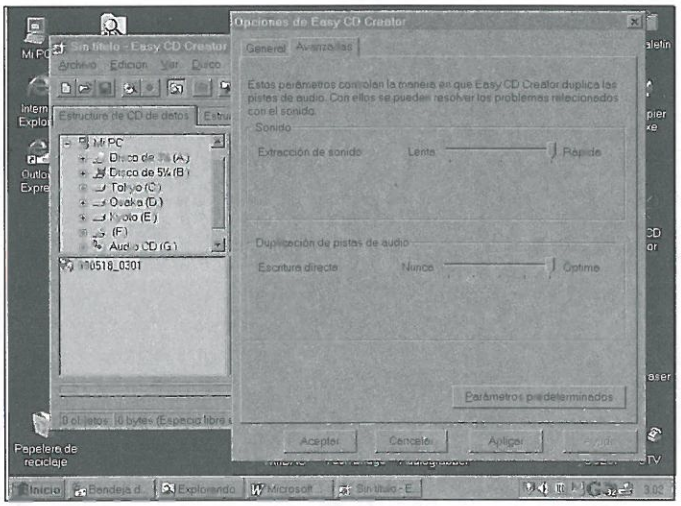
Sin ninguna duda, la opción más aconsejable para llevar a cabo tu interesante propuesta es utilizar Java, ya que a la hora de programar en un entorno cliente/servidor como el que propones es el más versátil; también tendrías la opción de programar en Visual Basic usando la tecnología Active Server Pages, pero para ello te recomendaríamos que te hicieses con una actualización de tu Visual Basic a la versión 5.0.

Desde luego la forma más sencilla de gestionar las comunicaciones es utilizar los recursos ya existentes en el propio Sistema Operativo, hemos de suponer que ya que usas Visual Basic, éste será el Windows 95 y su acceso telefónico a redes, simplificando considerablemente el trabajo con el hardware.

Lo siguiente a decidir es el protocolo a utilizar. Puedes elegir TCP/IP en el caso de utilizar Java, ya que existen multitud de tutoriales en Internet explicando la manera más adecuada de programar una arquitectura cliente/servidor. Te recomendamos que pruebes para empezar http://www.javasoft.com y consultes todo lo refererente a la programación de este lenguaje.

Si deseas conocer más profundamente el control del módem en Visual Basic en lo referente a su programación a más bajo nivel, tienes suerte puesto que en estos números de Sólo Programadores estamos publicando una serie de artículos que tratan ese mismo tema.

Resumiendo, te recomendamos Visual Basic o Java. Aunque puedes hacer uso de una mezcla de los dos creando el interfaz con Visual Basic 5 y aprovechando la potencia de éste en lo referente a su integración con applets Java. La elección definitiva dependerá de lo cómodo que te encuentres con cada uno de ellos.



Barra de opciones para enlentecer la extracción de audio.

Noticias frescas, cada día

TOWER
COMMUNICATIONS

EN INFORMÁTICA MARCAMOS EL CAMINO

SUPER NET MAGAZINE
PC MEDIA
SÓLO PROGRAMADORES
TOP PC PLAYER

CONTENIDO CD-ROM

Este mes el entorno de programación ADA 95 GNAT

PROGRAMACIÓN

- **ADA 95 GNAT v3.10** es un entorno de libre distribución de programación en ADA para diversos Sistemas Operativos. Allá por los años 60 y 70, el departamento de defensa de los Estados Unidos venía usando más de 2000 lenguajes para la programación de herramientas de misión crítica. La mayoría de estos eran lenguajes que fueron desarrollados para un objetivo específico. Finalmente en 1975, este departamento formó el HOLWG, o lo que es lo mismo, el grupo de trabajo de lenguajes de alto nivel, a fín de encontrar una solución para lo que se llamó, la "crisis del software".

  Lo que el HOLWG decidió, fue crear un lenguaje (otro), que fuese útil para todo, ya fuese programación, inteligencia artificial y lo más importante, programación en tiempo real de sistemas integrados. Esta programación es la usada para controlar los semáforos, misiles guiados y scanners de códigos de barras, computadoras de a bordo en los coches, aviones, equipos de música, etc.

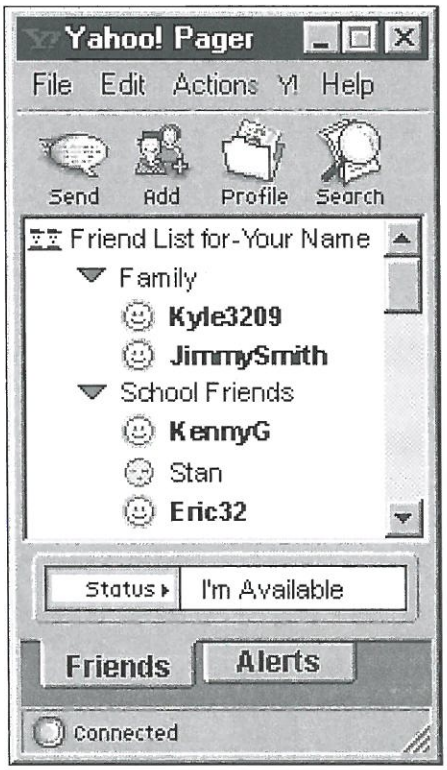
El nuevo lenguaje fue bautizado Ada, en memoria de la primera programadora de la historia

Más que crear ellos mismos el lenguaje, lo pusieron a concurso; simultáneamente todos los equipos de desarrollo crearon lenguajes basados en Pascal, al final el ganador fue Honeywell-Bull en Francia. El nuevo lenguaje fue bautizado Ada, en memoria de la que se dice que fue la primera programadora de la historia, Lady Ada Lovelace, hija del famoso Lord Byron y ayudante del también famoso matemático Charles Babbage que inventó la máquina analítica.

En 1983, el 'DoD' (Departamento de Defensa Americano), creó el primer estándar de Ada, llamado Ada 83; por aquel entonces, nadie podía crear un compilador de Ada sin la expresa autorización del 'DoD'.

Por el 1987 la cosa cambió, y en 1990 ya existían más de 200 compiladores válidos de Ada. En la actualidad, el estándar es el Ada 95, orientado a objetos y que ofrece interfaces para lenguajes como el C, FORTRAN y COBOL.

Aquí presentamos un entorno actual de programación para Ada 95, de libre distribución, el GNAT, o lo que es lo mismo, GNU/NYU Ada 95 de la Universidad de Nueva



Yahoo Pager permite contactar con amigos y familiares.

York, disponible para casi todas las plataformas.

NOTA: Todas las versiones para las plataformas más populares en forma de código fuente, salvo para Windows NT y OS/2, se encuentran en el directorio \PROG\GNAT del presente CDROM.

- **ObjectADA Real-time Raven** es un entorno de la empresa Aonix, proveedor líder en soluciones para entornos críticos desde 1989.

  Este entorno es especial sobre todo para aquellos programadores de aplicaciones que necesiten un funcionamiento seguro y tolerante a fallos, proporcionando tanto un kernel veloz, como una rica caja de herramientas para la creación de sistemas eficientes y deterministas en tiempo real de aplicaciones Ada 95 secuenciales o multitarea.

Con Developer/2000 se puede crear sencilla e instantáneamente aplicaciones cliente/servidor

- **Oracle Developer/2000 v2.0** es la aplicación de bandera de Oracle para el desarrollo de programas de trabajo en red. Con Developer/2000 los clientes pueden crear sencilla e instantáneamente aplicaciones cliente/servidor y de Web.

  Además, como novedad, incorpora un conjunto integrado de constructores para la creación de forms(formularios), informes, cuadros, consultas, objetos de bases de datos y procedimientos. La potencia de las capacidades declarativas de estos constructores permite a los programadores crear aplicaciones desde definiciones almacenadas sin escribir una sola línea de código. Adicionalmente, incorpora Wizards que automatizan la creación de una aplicación mediante una interfaz basado en el uso del puntero y el ratón.

  El entorno de desarrollo, la documentación en-línea y todo el resto del programa viene empaquetado en un único fichero zip de 176MB, lo cual es bastante engorroso pero su tamaño descomprimido abarcaba todo el contenido del CD-ROM. Así pues, debe ser descomprimido en un árbol de directorios antes de proceder a su instalación.

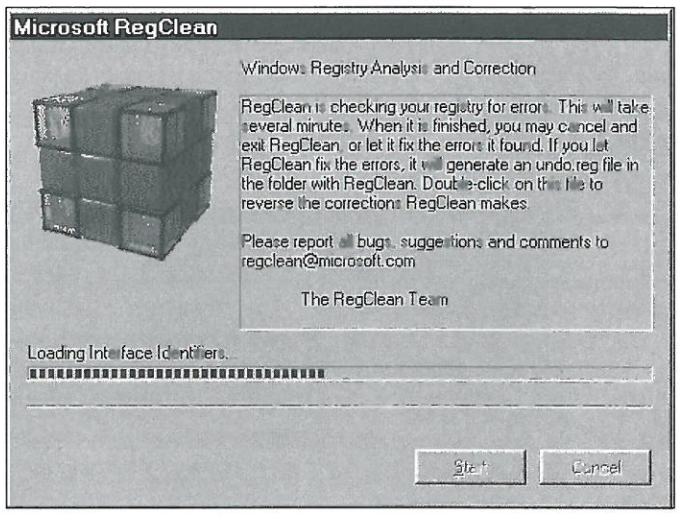
Yahoo Pager nos alerta de los nuevos mensajes.

REDES

- Las empresas afrontan actualmente un caos de administración. Cada conjunto de aplicaciones o sistema requiere sus propias herramientas de gestión, manejo de usuarios, seguridad e información de configuración en una base de datos propietaria. Esta implementación produce una administración redundante y aumenta los costes por el aprendizaje específico necesario.

Netscape Mission Control nos permite realizar un desarrollo de aplicaciones Internet reduciendo el riesgo de seguridad

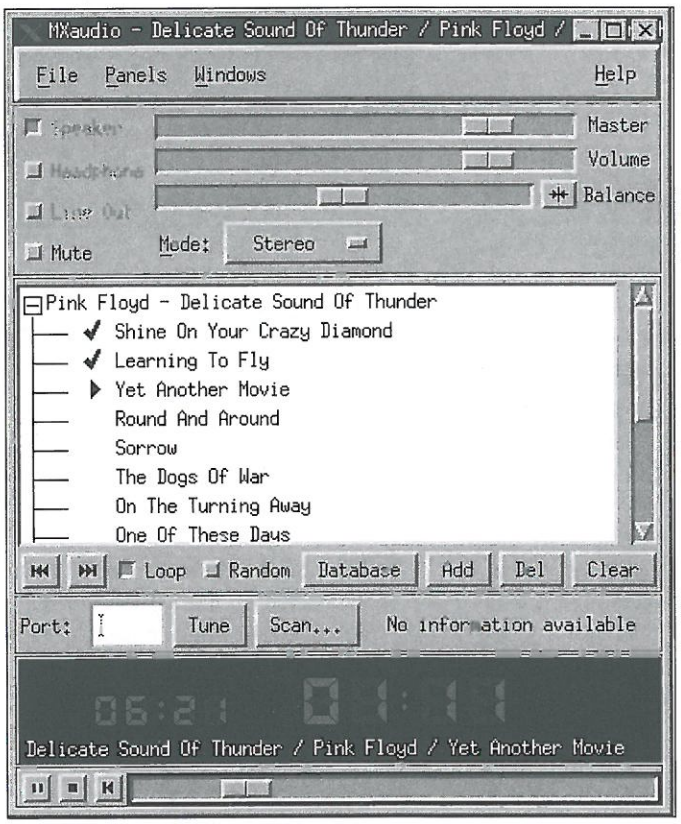
Netscape Directory Server v3.1, componente de la familia de productos Netscape Mission Control, soluciona estos problemas permitiendo un rápido desarrollo de aplicaciones Internet con unos servicios centralizados de administración, eliminando la gestión redundante de la red y reduciendo considerablemente el riesgo en lo referente a la seguridad.


Microsoft RegClean
Windows Registry Analysis and Correction
RegClean is checking your registry for errors. This will take several minutes. When it is finished, you may cancel and exit RegClean, or let it fix the errors it found. If you let RegClean fix the errors, it will generate an undo.reg file in the folder with RegClean. Double-click on this file to reverse the corrections RegClean makes.
Please report all bugs, suggestions and comments to regclean@microsoft.com
The RegClean Team
Loading Interface Identifiers...
Start
Cancel

Microsoft RegClean elimina los errores en el registro de Windows 95/NT.

- **Netscape Messaging Server v3.5** es un servidor de E-mail que reparte correo con sonidos, gráficos, ficheros de vídeo, forms HTML, applets Java y aplicaciones. Messaging Server se caracteriza además por una gran velocidad de proceso de los mensajes, manejo de colas y alta potencia en los accesos a directorios. Las organizaciones que usen este servidor, podrán comunicarse prácticamente con la totalidad de sistemas de correo y pasarelas.

Messaging Server se caracteriza por una gran velocidad de proceso de los mensajes y alta potencia



Xaudio V2.5 es una implentación de la librería de decodificación de audio MPEG.

Messaging Server puede ser actualizado con información procedente de Directory Server. Además, ofrece la posibilidad de acceder fácilmente a todos los directorios y libretas de direcciones. Posee una gran sencillez para añadir nuevas aplicaciones con características especiales, como el control de flujo en el tráfico saliente y la conversión de juegos de caracteres, a fín de asegurar la absoluta compatibilidad con otros idiomas.

- **Yahoo Pager (build 365)** le permite a uno mantenerse en contacto con sus amigos, familiares y colegas. Con Yahoo Pager se puede saber cuándo nuestros amigos están disponibles para hablar con ellos o simplemente si están conectados a la red.

Podremos enviar mensajes instantáneos y privados, chatear con amigos en intimidad o simplemente con pulsar Send, escribirles un mensaje rápido.

Yahoo Pager además nos notifica la llegada de nuevo correo inmediatamente, sin necesidad de abrir nuestro lector de E-mail y comprobar si tenemos algún mensaje.

Una vez que lo hayamos puesto en marcha, se nos ofrecerá la opción de invitar a un amigo a que se instale este mismo programa, buscar nuevos amigos en la red que ya lo tengan instalado o conectarnos con el centro de ayuda Yahoo para consultar posibles dudas que nos pudieran surgir.

Yahoo Pager además nos notifica la llegada de nuevo correo inmediatamente

Xaudio V2.5 se usa con distintos GUI o entornos gráficos.

- **X-Win32 v4.01** nos va a permitir ejecutar aplicaciones X-Windows en nuestro entorno Windows 95/NT como si lo hiciéramos localmente.

  Lo único que tenemos que hacer es arrancar el software servidor en la máquina Unix, por ejemplo Linux. Para ello, lo más sencillo es hacer un telnet a la máquina remota a través de nuestra red TCP/IP, ejecutar el servidor X-Windows (software servidor) con xdm, y seguidamente ya en un entorno gráfico, se nos solicitará el nombre de usuario y contraseña. De esta manera, nos será posible visualizar cualquier aplicación Unix gráfica en nuestro ordenador cliente pero ejecutada remótamente en el servidor Unix.

ObjectManual es una herramienta de documentación en-línea en formato HTML, MIF y RTF

  La única limitación de la versión demo es el tiempo limitado de conexión al servidor, que es de dos horas por sesión.

UNIX

- Una buena documentación para una aplicación software es algo no habitual. El éxito o fracaso de un programa no sólo esta gobernado por su funcionalidad, sino también por la calidad de la documentación que le acompañe.

  Los proyectos software sufren de ajustes en el tiempo y el presupuesto. Una creciente popularidad entre el software aflorante es proporcionar una documentación en-linea más que en papel. Esta documentación se facilita en la forma de hiper-texto o hiper-enlaces y puede ser convertida en papel cuando se solicite.

Menú del sistema Xaudio V2.5.

X-Win32 v4.01 nos va a permitir ejecutar aplicaciones X-Windows en nuestro entorno Windows 95/NT

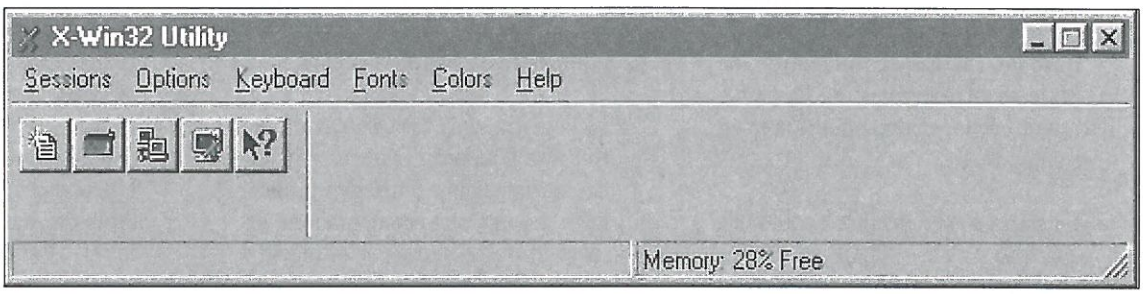
  ObjectManual es una herramienta automatizada para la generación de documentación en-línea en formato HTML, MIF y RTF. Esta herramienta puede generar documentación para clases C++ creando documentos HTML que puedan ser navegados por cualquier browser HTML, además de permitir ser editada de la misma manera.

- **XAudio v2.5** fue escrito para ser una rápida implentación de la librería de decodificación de audio MPEG, a fín de ser usada con diferentes GUI o entornos gráficos.

  La librería del decodificador permite ser licenciada, tanto las fuentes como los ejecutables.

  Hemos incluido también el software de desarrollo para cada una de las plataformas. Las principales características de esta librería son:

  - MPEG Audio Layer I, II y III (ficheros .mp1, .mp2, .mp3).
  - Diferentes bitrates, soporta Mpeg 1 y 2.
  - Soporte para dispositivos genéricos de audio.
  - Módulos de entrada/salida.
  - Control del decodificador síncrono y asíncrono.
  - Optimización del código C (30% más rápido).
  - Optimización Pentium.
  - Probado en plataformas Intelx86, Sparc, PowerPC y Mips.
  - Acceso rápido aleatorio al flujo de datos (incluso en Layer III).

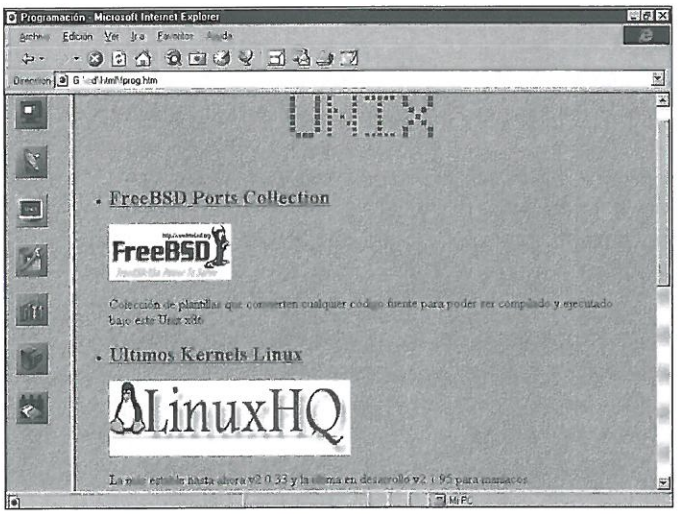


Con X-Win 32 V4.01 podremos visualizar cualquier aplicación Unix gráfica.

- Control del audio (mono, estéreo, volumen, balance, etc.)
- Admite la decodificación de varios flujos de datos simultáneos.
- Robusto.

La librería se combina con un driver de los siguientes soportados:

- FreeBSD.
- Linux.
- Solaris (Sparc y x86).
- Irix.
- BeOS.
- SCO Openserver 5.



A través del interfaz HTLM es posible acceder a las colecciones de elementos Unix.

XINU v1.0 es un Sistema creado para enseñar el funcionamiento de un sistema multitarea en tiempo real

Finalmente necesitamos un interfaz para el usuario. Existe uno en modo texto (en línea de comandos) para todas las versiones y otro gráfico para las siguientes:

- Motif (Linux,Solaris e Irix)
- BeOS

NOTA: Todas las versiones así como sus SDKs para las plataformas más populares se encuentran en el directorio \UNIX\XAUDIO del presente CDROM.

- **XINU v1.0** es un Sistema Operativo creado en la Universidad de Canberra en Australia, para enseñar a los estudiantes de Sistemas Operativos el funcionamiento de un sistema multitarea en tiempo real.

  Es capaz de ser ejecutado como una aplicación más en DOS o en una ventana tanto de Windows 95 como de Windows 3.11 para trabajo en grupo y Windows 3.1.

  Además facilitamos un SDK para recompilar el entorno a nuestro gusto.

RegClean está diseñado para limpiar las entradas innecesarias

NOTA: Todas las versiones para las plataformas más populares así como documentación al respecto se encuentran en el directorio \UNIX\XINU del presente CDROM.

VARIOS

- **Microsoft RegClean v4.1ª** es un programa orientado para eliminar los errores que pudieran existir en el Registro de Windows 95/NT.

  El Registro es la base de datos central para Windows. Cuando se instalan, desinstalan y reinstalan programas, en nuestra computadora se crean entradas nuevas en el registro y otras se modifican o eliminan. A lo largo del tiempo, nuestro registro del sistema se corrompe, desaprovecha e inhabilita en parte. Como resultado, empieza a dar problemas usando OLE o utilizando los accesos directos.

  RegClean está diseñado para limpiar las entradas innecesarias, haciendo que el sistema vaya más rápido. Atención: No arregla registros defectuosos, así que aconsejamos realizar un backup del mismo periódicamente. La versión proporcionada aquí funciona a la perfección teniendo instalado el Internet Explorer 4.0.

WOCAR puede procesar también cualquier imagen TIFF bi-nivel.

- **Wocar v2.5** es una aplicación para el reconocimiento óptico de caracteres (OCR). Convierte documentos escaneados en documentos de texto. El software procesa documentos escritos en Inglés y Francés. WOCAR puede funcionar con cualquier escáner que soporte la interfaz TWAIN. Puede procesar también cualquier imagen TIFF bi-nivel. Esta aplicación funciona tanto en Windows 95 como en NT.

A PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SOLO PROGRAMADORES

IV, NÚMERO 46

975 Ptas.

NA OPCIÓN DE GESTIÓN POTENTE Y VELOZ

POSTGRES
LA BASE DE DATOS DE LINUX

TECNOLOGÍA 3D
Programación del acelerador gráfico i740

WINDOWS NT SERVER
Acceso remoto (RAS)

VISUAL BASIC
ActiveX unido a controles de datos

PROGRAMACIÓN MULTIMEDIA
DirectX con Delphi: DirectPlay

HARDWARE
Programación del módem (III)

PROGRAMACIÓN DE SOCKETS
Creación de un servidor Web con Delphi (II)

CONTENIDO DEL CD-ROM

- Netscape Messaging Server 3.5
- Netscape Directory Server 3.1
- Oracle Developer 2000 2.0
- ADA 95 GNAT 3.10
- ObjectADA Realtime Raven
- X-Win32

8 413042 303299